Debreceni Egyetem
Informatikai Kar

Adatbázis-alapú alkalmazás-fejlesztés Borland Delphiben

Készletnyilvántartó program

Témavezető:
Dr. Bajalinov Erik
tudományos főmunkatárs

Készítette:
Zakor Szilárd
programozó matematikus

Debrecen

Tartalomjegyzék

Köszönetnyilvánítás:

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Bajalinov Eriknek a szakdolgozatom elkészítésében nyújtott segítségéért és útmutató tanácsaiért. Köszönöm a feleségemnek, Szilágyi Erzsébetnek és a kislányomnak, Zakor Dórának azt a sok türelmet, amely nélkül e dolgozatom nem készülhetett volna el.

1. BEVEZETÉS

Napjainkban egyre többször akarva akaratlanul belebotlunk a számítógépekbe. Ez az a dolog, ami manapság már szinte közhely, de ezt a világot éljük. Lassan az élet szinte minden területén szolgálni fognak minket. A számítástechnika és informatika területén tapasztalható gyors fejlődés ma óriási lehetőségeket biztosít a rendszerfejlesztők, programozók és ezen keresztül az felhasználók számára. A szoftverek túlnyomó többsége arra hivatott, hogy elősegítse, meggyorsítsa, megkönnyítse és biztonságosabbá tegye az emberek, cégek munkáját. A szakdolgozatom elkészítésénél ezek a célok vezéreltek, valamint a Delphi fejlesztői környezet és ezen belül a Delphi adatbázis kezelésének a bemutatása.

A témaválasztásom a Keleti Nap Bt. ruházati boltjának egyedi kód szerinti készletnyilvántartására esett, mivel itt szükségessé vált a régi manuális és sokszor követhetetlen nyilvántartásról való áttérés egy hatékonyabb és gyorsabb számítógépes regisztrációra.

1.1. A Keleti Nap Bt. bemutatása

Ez a kisvállalkozás 1994-ben jött létre. A jó kapcsolat a Magyarországon élő kínai nagykereskedőkkel, valamint a szerető feleség, aki szintén kínai születésű, arra késztette a jelenlegi tulajdonost, hogy elindítsa ezt a gyümölcsöző üzletet. A beindulás folyamán csak egy kisebb üzletrészben folyt a bolti tevékenység, de mihamarabb rá kellett döbbenni, hogy az áruk utáni nagy kereslet miatt nagyobb üzletbe szükséges áthelyezni ezen tevékenységet. A költözés alkalmával derült fény arra, hogy egy pontos, precíz nyilvántartásra lenne szükség.

Jelenleg családi vállalkozásként folyik az üzleti tevékenység, de a nagy piaci kereslet előbb-utóbb arra készteti a cégvezetőt, hogy tovább bővítse kereskedelmi hálózatát. Alkalmazottak felvétele, valamint újabb kereskedés megnyitása válik

szükségessé. Mint ebből is már kiderült, egy kínai ruházati bolt portékáinak naprakész állapotának tükrét kell megvalósítani.

Papír alapon folyik a termékek nyilvántartása, és ez nagyon sok esetben káoszhoz vezet, hiszen a bővülő termékskála mind bonyolultabbá teszi az árucikkek regisztrációját. A különböző hatósági szervek ellenőrzésekor csak nagy és fáradtságos munka árán lehetett a valóságos képet kirajzolni. Szerencsére még soha nem adódott semmilyen bonyodalom, de ezek mihamarabbi elkerülése végett egy pontos, precíz, gyors rendszer létrehozása vált szükségessé. Remélhetőleg az elkészült szoftver nagy hasznára lesz a tulajdonosnak, hisz eddig nagy erőfeszítések árán lehetett csak rendet tartani a vállalkozás ezen területén.

Ezen a programon keresztül mutatnám be a Delphi adatbázis kezelését, valamint a Delphi adatbázis-kezelő főbb komponenseit.

2. A PROGRAMOZÁSI NYELVRŐL

A mai rohanó világunkban az egyik legnagyobb probléma az állandó időhiány. A programozóknak ugyanannyi vagy talán kevesebb idő alatt kell sokkal szemléletesebb, barátságosabb és természetesen az elvárásoknak megfelelően működő, stabil, megbízható alkalmazásokat készíteniük. Talán emiatt terjedt el egyre több helyen az objektum-orientált szemléletmód. Egyre nagyobb teret hódítanak a vizuális negyedik generációs fejlesztőeszközök is, melyekben adatbázis-specifikáló, képernyőtervező és jelentéstervező eszközök segítségével lényegesen kevesebb kódolással készíthetjük el alkalmazásainkat.

A Delphi a Borland Software Corporation cég Windows grafikus felületen futó Object Pascal alapú negyedik generációs (4GL) programozási nyelve. Olyan gyors alkalmazásfejlesztő környezet (RAD, Rapid Application Development), amely hazánkban rendkívüli népszerűségnek örvend. A Delphi szerencsésen egyesíti magában a Windows és a Linux alkalmazások készítésére szolgáló grafikus fejlesztői környezetet és a teljesen objektum-orientált programnyelvi fordítót. A rendszer támogatja a legújabb

technológiákat, az Internet-alkalmazások készítését, a multimédiás megjelenítést, az ügyfél-kiszolgáló adatbázis-kezelést. Ha csupán a főbb tulajdonságokat tekintjük, azt találjuk, hogy a Delphi az egyik leghatékonyabb alkalmazásfejlesztő eszköz, amely Object Pascal alapú forrásnyelvi fordítóprogrammal, komponens alapú felépítéssel, és testre szabható adatbázis hozzáférési lehetőséggel rendelkezik. A rendszer része egy másik komponenskönyvtár is, mellyel Windows környezetben Linux alkalmazást fejleszthetünk. A CLX-re alapozott forrásnyelvű programból a Linux rendszerben működő Kylix fordító segítségével Linux alkalmazást készíthetünk.

A Delphi alapját a Vizuális Komponensek Könyvtára (Visual Component Library) képezi. Valójában egy vizuális programozási nyelv, amely magas szinten használja az objektum-orientált keretrendszert. Ahhoz, hogy jó alkalmazói programokat készítsünk, vagy komponenseket fejlesszünk, nélkülözhetetlen a Delphi alapos ismerete. Ezzel a fejlesztőeszközzel nagyon gyorsan és hatékonyan tudunk adatbázis-kezelő alkalmazásokat fejleszteni. Kiemelkedően támogatja az adatok kezelését.

3. AZ ADATBÁZISOKRÓL RÖVIDEN

Adatbázison köznapi értelemben valamely rendezett, valamilyen szisztéma szerint tárolt adatokat értünk, melyek nem feltétlenül számítógépen kerülnek tárolásra. Az adathalmaz csak akkor válik adatbázissá, ha az valamilyen rend szerint épül fel, mely lehetővé teszi az adatok értelmes kezelését. Természetesen ugyanazon adathalmazból többféle rendszerezés alapján alakíthatunk ki adatbázist. Az adatok tárolásába bevitt rendszernek alkalmasnak kell lennie a leggyakrabban előforduló igények hatékony kielégítésére. Az adatbázisok mellé egy adatbázis-kezelő rendszer (DBMS) is járul, mely az adatbázis vagy adatbázisok üzemeltetését biztosítja. Hagyományos adatbázis esetén ez a kezelőszemélyzet intelligenciájának része, elektronikus adatbázisok esetén pedig valamilyen szoftver.

A Delphi rendszer egyik kiemelkedő erőssége a különböző típusú adatbázisok kezeléséhez szükséges technológiák és eszközök sokoldalú támogatása. A napjainkra kialakult adatbázis-modellek közül a Delphi a legelterjedtebb modellt, a relációs

adatbázisok kezelését támogatja. A relációs adatbázis alapvető fogalma a táblázat (adattábla, table), amely egy előre meghatározott számú oszlopot, adatmezőt (field) tartalmaz. Az adatmezők értékeit a táblázat sorai, más néven bejegyzései vagy rekordjai (record) tartalmazzák. Az adattáblák azonos szerkezettel rendelkező rekordjaiban tárolt adatokat a megfelelő mezőnév megjelölésével egyenként érhetjük el. Az adatbázis relációs volta abból adódik, hogy több táblázat létrehozása esetén a bennük tárolt adatok között különböző kapcsolatokat (relációkat) írhatunk le. Így az egyik táblázatban megtalált adat alapján egy másik táblázatban is megkereshetjük a megfelelő bejegyzést.

Az adatbázis-műveltek többsége valamilyen lekérdezés jellegű művelet, lényegében a számunkra információt hordozó bejegyzések kiválasztása a tárolt adatok közül. Az adatlekérés elvégzésére a relációs adatbázis-kezelő rendszerek gyakran az SQL-t (Structured Query Language), a strukturált lekérdező nyelvet használják. Bár az SQL nyelvnek létezik szabványosított változata, a legtöbb rendszer mégis saját nyelvjárást használ, kiegészítve, módosítva a szabványt.

Az adatkeresés egy másik, rendkívül gyors módszere az indexelés, amely a Paradox, a dBASE, és az SQL táblákban egyaránt alkalmazható, bár kissé eltérő módon. Az adattábla strukturálásakor létrehozott indexek, illetve indextábla technikailag egy olyan fájl, amely tartalmazza az adatmezők elérési sorrendjét. Ennek köszönhetően az indexek használatának egyik nagy előnye, hogy eleve egy bizonyos szempont szerint rendezett adattáblával dolgozhatunk. A relációs adatbázisok kezelése során az adatbázis-kezelő rendszer belső mutatója (cursor) mindig az elért rekord aktuális mezőjére hivatkozik. Természetesen minden időpillanatban pontosan egy rekord, illetve e rekordon belül egy mező lehet aktuális.

Egy adatbázist kétféleképpen is azonosíthatunk. Megadhatjuk annak a könyvtárnak a teljes elérési útvonallal kiegészített nevét, amely az adattáblákat tartalmazza. A másik, a Delphi rendszer által javasolt módszer szerint az adatbázisunkhoz logikai (alias) nevet definiálhatunk megfelelő segédprogramok segítségével.

4. AZ ADATBÁZIS – KEZELÉSI ARCHITEKTÚRÁK

Minden adatbázis-kezelő alkalmazásban három fő funkcionális egységet különböztetünk meg:

- ***A közvetlen adatkezelés (Data processing)***

  Az alkalmazásnak ez a része végzi el a tárolt adatok fizikai feldolgozását: állományok nyitása, zárása, indexelések, a lekérdezések optimalizálása és futtatása, új adatok felvitele, meglévők törlése, módosítása, az adatok cache-elése, a zárolási konfliktus-helyzetek feloldása. Ennek a résznek a megvalósítása függ az adatok tárolási módjától, azaz a használt adatbázis formátumától.

- ***Az alkalmazás-logika (Business Logic)***

  Ez a rész felelős a teljes alkalmazás helyes működéséért. Biztosítja az adatok védelmét (felhasználói jogosultságok), elronthatatlanságát (integritását), hatékony és kényelmes kezelését.

- ***A felhasználói felület (User Interface)***

  Ez a rész a felhasználóval való közvetlen kapcsolattartásért felelős. A felületnek minél tetszetősebbnek, barátságosabbnak, és ugyanakkor elronthatatlannak kell lennie. Az elronthatatlanság alatt itt a felhasználói felület elemeinek helyes működésére gondolunk. Az adatok helyességéért nem a felhasználói felület, hanem az alkalmazás-logika felel.

Ehhez a három részhez még hozzátartozik egy utolsó, a tényleges tárolt adat, azaz a fizikai adatbázis. Az alkalmazásnak azonban a konkrét adatok nem képezik részét.

4.1. Adatbázis architektúrák

- ***Egygépes megvalósítás (Local Databases)***

  Az adatbázisoknak ez a lehető legegyszerűbb megvalósítási módja. Az alkalmazás egyetlen gépre íródott, az adatbázis és az azt feldolgozó program ugyanazon a gépen helyezkedik el, az adatbázist csak egyetlen program használja egy időben. Ebben az esetben mindhárom réteg egyazon gépen helyezkedik el.

- ***File-kiszolgáló (File-Server) architektúra***

  Ebben az esetben az adatbázis állományok átkerülnek egy központi szerverre és egyidőben több program is használhatja őket a hálózaton keresztül. A szerver csak az adatok tárolására szolgál. Ezen megoldás esetén, ha a felhasználó akármilyen egyszerű adatműveletet akar is végrehajtani, az adatrekordoknak el kellett jutniuk a felhasználóhoz a hálózaton. Ez nagy adatforgalommal jár, ami a hálózat túlterheléséhez vezethet. Szintén mindhárom fentebb tárgyalt réteg egyazon gépen helyezkedik el.

- ***Ügyfél-kiszolgáló (Client/Server) architektúra***

  Az adatbázisok implementálásának e formájában az alkalmazás két részre bomlik. Az adatok közvetlen kezeléséért egy adatbázis-szervernek nevezett software a felelős, míg a felhasználóval való kapcsolattartás az ügyfél program feladata. A client-server technológiában az ügyfél utasítja a szervert, pl. adatokat kér le, és erre a szerver visszaküldi az eredményt. Tehát nem kell a hálózaton a feldolgozandó adatoknak rekordról-rekordra átmenni a klienshez, hanem egy rövid parancs hatására, csak a ténylegesen kért, hasznos adatok fognak a szervertől a kliensig utazni, ezáltal jelentősen csökkentve a hálózati forgalmat. Így az adatfeldolgozást a szerver végzi a kliens parancsainak hatására. E parancsok számára kidolgoztak egy szabványos nyelvet, ez az SQL.

- ***Több rétegű (Multi-Tier) adatbázis architektúra***

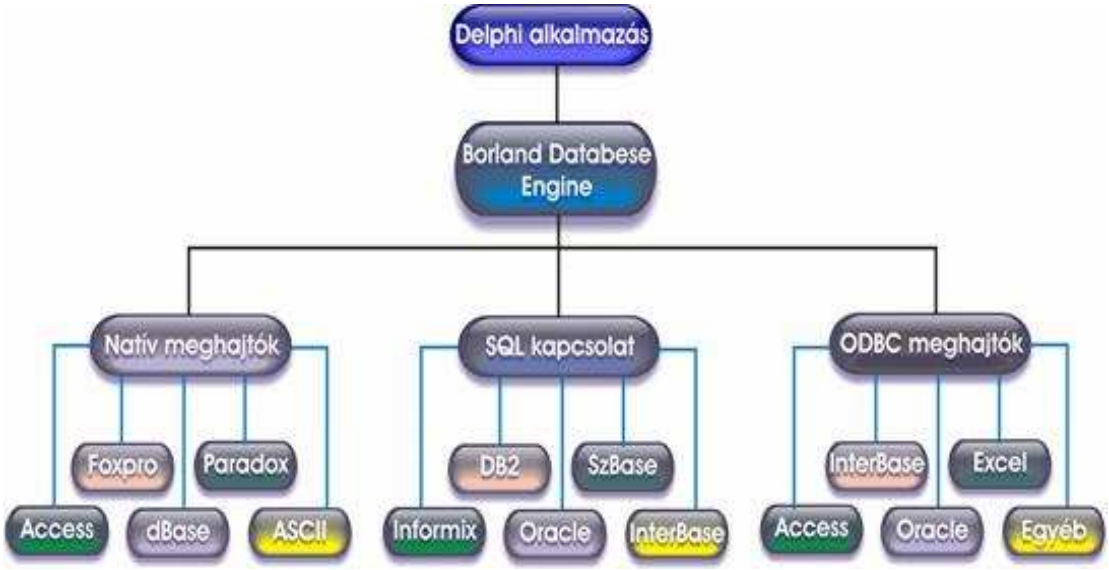
  Ebben az esetben a kliens nem közvetlenül az adatbázis-szerverhez, hanem egy vagy több köztes, ún. applikációs szerverhez kapcsolódik, és végül az applikációs szerver kapcsolódik az adatbázis-szerverhez. Tehát a kliensek a középen elhelyezkedő applikációs szervertől kapják az adatokat, ezért ezt a réteget adatszolgáltatónak (Data Broker) is nevezik. Így az adatbázis-logikát el lehet helyezni a középső rétegben, és a kliens feladata csak a felhasználóval való kapcsolattartás lesz. Az ilyen kliens-t „sovány” (thin) kliensnek nevezzük, hiszen a munka nagy részét az applikációs szerver végzi. Tehát ebben az esetben a három réteg fizikailag is három különböző helyen helyezkedhet el.

A Delphiben lehetőségünk van a fentebb említett bármelyik architektúrát felhasználva adatbázisos alkalmazást készíteni. Az adatbázisok kezelése speciális komponensek segítségével történik. Alkalmazásainkban a különböző formátumú adatbázisokat egységesen, ugyanazokkal a komponensekkel érjük el. A komponensek metódusai a beépített adatbázismotor (Borland Database Engine, BDE) rutinjait használják. Tehát a BDE egy egységes felületet (rutincsomagot) biztosít a különböző formátumú adatbázisok Delphiből történő kezelésére.

5. BORLAND DATABASE ENGINE (BDE)

A BDE a Borland Database Engine rövidítése, amely egy Windows alapú 32 bites adatbázis-motor. Feladata, hogy kapcsolatot teremtsen a fizikai adatbázis és az azt kezelő alkalmazások között, és ezzel megkönnyítse a Borland fejlesztőrendszereit használó programozók munkáját. Leveszi a vállukról a táblaformátumok kezelését, és alapvető adatbázis-kezelő rutinokat tartalmaz. A Delphi adatbázis-kezelése teljes mértékben a BDE motorra épül, a Delphi minden DB komponense ezt használja az adatok közvetlen vagy közvetett eléréséhez. A BDE API közel 200 eljárást és függvényt tartalmaz, de közvetlen hívásukra az alkalmazások többségében csak igen ritkán kerül sor. Ehelyett a BDE használat általában a Delphi adatelérési vezérlőelemein keresztül

történik, amelyek az egyszerű kezelhetőség mellett, a BDE API hívásainak majdnem teljes palettáját megvalósítják. A BDE függvényei objektum-orientáltan épülnek fel, így azok elérése egyszerű és strukturált. A programozási felülete egy sor olyan feladat elvégzése alól mentesíti az adatbázis-kezelő alkalmazások programozóit, mint az adattáblák és bejegyzések zárolása, a bejegyzések frissítése, alap I/O műveletek stb. Ennek köszönhetően a fejlesztő több figyelmet szentelhet magának az adatnak és a vele elvégezendő műveleteknek, megszabadulva az adatelérés apró részleteitől. Technikailag a BDE az adatbázis eléréshez szükséges DLL-ek gyűjteménye. Telepítése után párhuzamosan több alkalmazás is osztozhat rajta. Mivel előre nem tudhatjuk, milyen



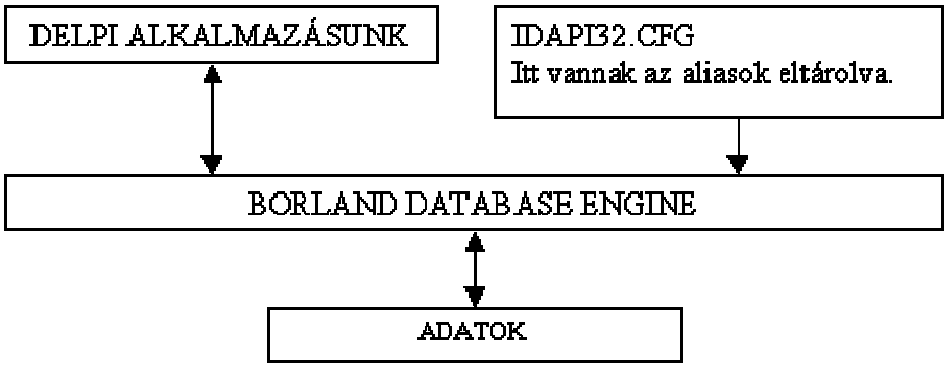
1. ábra

szoftverfelszereltségű gépen fogják futtatni a kész programunkat, az alkalmazásunk telepítő csomagjának mindig tartalmaznia kell a BDE-t is. Ha az alkalmazásunk írásakor a BDE lehetőségeit használjuk, mind a gépünk lokális, például dBASE, Paradox adattábláit, mind a távoli adatbázis-kiszolgálón lévő, illetve az ODBC (Open DataBase Connectivity) illesztőprogramok által támogatott fájlokat is könnyen elérhetjük.

A BDE fontos szolgáltatása még az ún. alias-ok kezelése. Az alias-ok segítségével elnevezhetjük adatbázisunkat, hogy később ezen a néven hivatkozzunk rá annak fizikai helyétől függetlenül. Standard adatbázisoknál az alias gyakorlatilag egy egyszerű mutató, ami egy adott könyvtárra mutat.

5.1. BDE aliasok

A Borland adatbázis motor álneveket (alias) használ a különböző adatbázisokra való hivatkozáskor. Az alias gyakorlatilag paraméterek halmaza, ami egyszerűbb esetben lokális adatbázisoknál az adatbázis elhelyezkedését és típusát tartalmazza, adatbázisszerverek esetén pedig plusz paramétereket, amelyek a szerverhez való csatlakozáshoz szükségesek. Amikor elkészítjük az alkalmazásunkat, akkor abban álnevekkel hivatkozunk a használt adatbázisra. Így, ha később például megváltozik az adatok elérési útvonala, az nincs fixen belefordítva a programunkba, hanem egyszerűen megváltoztathatjuk azt a későbbiek során bármikor. Alias-t a BDE Administrator-ral, a Database Explorer-rel, de akár saját magunk programból is létrehozhatunk. A létrehozott alias a BDE saját konfigurációs állományában kerül elmentésre, és



2. ábra

mindaddig megmarad, míg nem töröljük. Miután elindítottuk a BDE Administratort, már alapesetben is látható lesz néhány alias, ettől nem kell megijedni, ezek példák, melyeket a Delphi hoz létre amikor felinstalláljuk, és a saját példa adatbázisaira mutatnak.

6. A DELPHI NÉHÁNY ADATBÁZIS-KEZELŐ KOMPONENSE

A Delphi több adatbázisokhoz kapcsolódó komponenssel is rendelkezik. A komponenspaletta adatelérés (Data Access) lapján találhatjuk azokat az összetevőket, amelyek segítségével a BDE - központú adatbázisokat használhatjuk. Legtöbbjük nem

látható komponens, hiszen adatkapcsolatokat, táblákat, lekérdezéseket, vagy hasonló elemeket tartalmaznak. Szerencsére a Delphi számos olyan eszközt is tartalmaz, amelyekkel láthatóvá, és ezáltal közvetlenül szerkeszthetővé tehetjük adatbázisainkat. Ezeket az elemeket a Data Controls lap biztosítja számunkra. Az itt található elemeket adatfüggő vezérlőknek nevezzük.

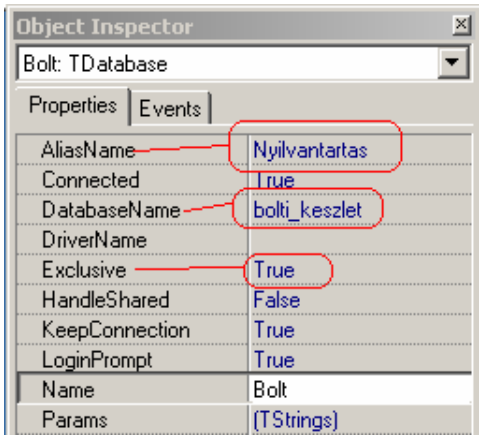
A felsorolásomban nem térek ki mindegyik komponensre, csak a fontosabbakat említem meg.

6.1. Data Access komponensek

Ide azok a komponensek tartoznak, amelyek segítségével megvalósítjuk az adathozzáférést, az adatbázis fájljaival való kapcsolatot, ezek a komponensek kapcsolják össze az adatokat az adatmegjelenítő, manipuláló objektumokkal.

- ***Database komponens***

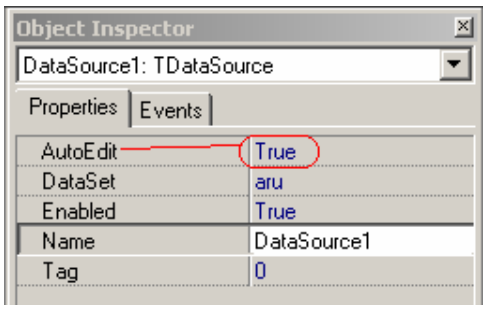
  A Database komponens egy konkrét adatbázis elérését biztosítja. Az adatbázis-szerverekhez való csatlakozáshoz szükség van egy ilyen objektumra, ez biztosítja az utat az adatbázis felé, csatlakozáskor ez felelős a jelszó bekéréséért, metódusaival tranzakció-kezelést tudunk lebonyolítani. Ha Database komponenst használunk, akkor a tábla-, lekérdezés- stb. komponensek a DatabaseName jellemzőjükkel nem az álnévre fognak hivatkozni, hanem az adatbázis-komponensre, az adatbázis-komponens pedig az AliasName jellemzőjével rámutat majd az adatok álnevére.



3. ábra

- ***DataSource komponens***

A Delphiben egy adatbázis eléréséhez mindenekelőtt szükségünk van egy adatforrásra, amelyet a DataSource komponenssel adhatunk meg. Ez a komponens teremt kapcsolatot az ún. DataSet komponensek és a vizuális adatmegjelenítő (manipuláló) komponensek között. A DataSet komponensek azok a komponensek, amelyek valamilyen adathalmazt biztosítanak (TTable, TQurey). A komponens DataSet tulajdonságát kell a megfelelő komponensre (Table-re vagy Query-re) állítani.



4. ábra

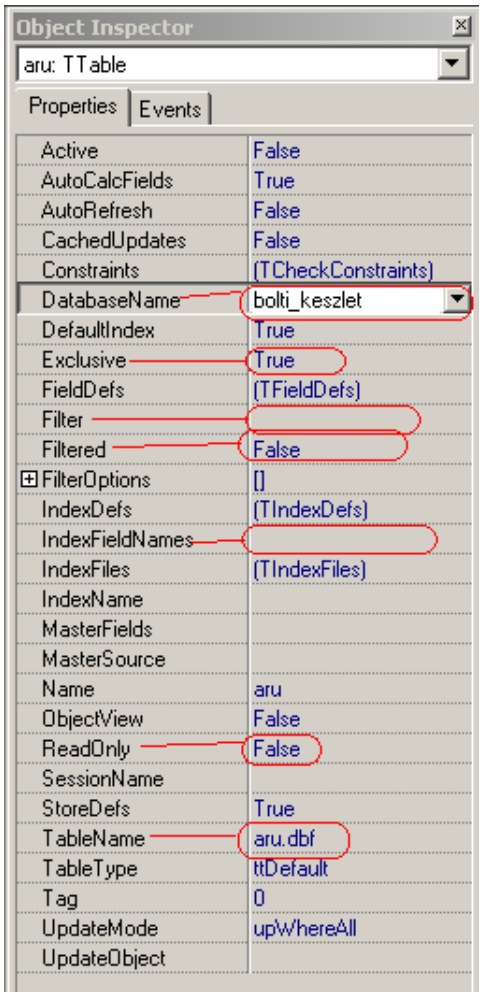
A DataSouce vezérlőelem AutoEdit tulajdonságának értékét akkor érdemes átállítani false-ra, ha el szeretnénk kerülni az objektum Edit() metódusának automatikus meghívását. Az automatikus meghívás akkor történik, amikor egy, az adatforráshoz a DataSource vezérlőelemen keresztül hozzákapcsolt adat megjelenítő vezérlőelem megkapja a fókuszt, így ebben az üzemmódban a felhasználói változtatás az adatbázisra is hatással lesz. Az Enabled tulajdonság false-ra való beállításával elérjük, hogy a DataSource vezérlőelemhez kapcsolódó adatmegjelenítő vezérlőből „eltűnik” a megjelenítendő adat. A DataSource objektum State tulajdonságában tárolt érték tesztelésével az alkalmazásunk tudomást szerezhet az adatforrás vezérlőelem (DataSet) aktuális állapotáról: aktív-e vagy sem, illetve milyen adatbázis-művelet végrehajtása folyik éppen.

A TDataSource osztály három saját eseményt is definiál. Az OnDataChange az adatbázis aktuális rekordjának változtatása által kiváltott

esemény, az OnStateChange és az OnUpdateData események rendre az adatforrás-vezérlőelem State tulajdonságában tárolt érték változtatását, illetve az adatbázis aktuális rekordjának frissítését jelzik.

- ***Table komponens***

A legegyszerűbb módszer, amivel adathozzáférést biztosíthatunk a Delphiben, a Table komponens használata. A Table objektumok egyszerűen egy adatbázistáblára hivatkoznak. A segítségével beépíthetünk egy táblát a programunkba. A komponens a BDE-t használva hozzáférést biztosít egy adatbázis-táblához. Ha felhelyezünk egy ilyen komponenst a form-ra, az első dolog, amit tennünk kell, a táblázat fájlnevének megadása. Ehhez először a DatabaseName-et kell megadnunk, amennyiben létrehoztunk az adatbázisunk



5. ábra

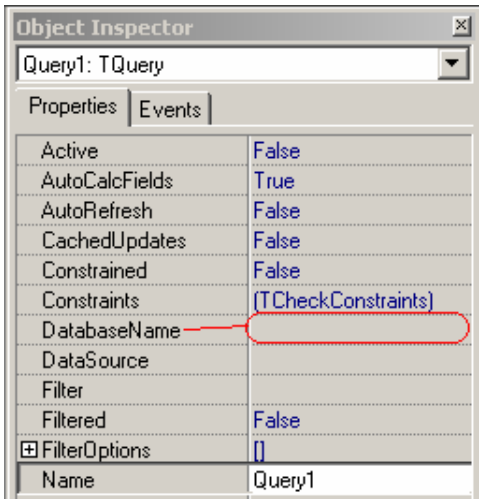
részére alias-t. Ezt a TableName tulajdonság beállítása követi. A lenyíló listában megtalálhatjuk az adatbázisunk fájljait, vagy ha nem használunk alias-t, a program könyvtárában lévő fájlokat.

Ahhoz, hogy a felhasználó ne tudja módosítani az adatbázisban tárolt adatokat, true értékre kell állítanunk a Table vezérlőelem ReadOnly tulajdonságát. Az Exclusive tulajdonság értékének true-ra való állításával pedig letilthatjuk az adattábla más alkalmazások általi módosítását arra az időre, amíg a táblát a programunk használja.

A Table vezérlőelem az adattáblán belüli bejegyzések egy csoportját is képviselheti a különböző indexek és szűrők használata esetén. A szűrő karaktersorokat a Filter tulajdonság értékeként kell megadni, majd engedélyezni kell a szűrést a Filetered tulajdonság true-ra való beállításával. A rekordok megjelenítési sorrendjét módosíthatjuk, ha a Table vezérlőelem IndexFieldName tulajdonsága értékeként megadjuk a rendezéshez használható mező nevét.

- ***Query komponens***

Ez a komponens egy olyan DataSet-et épít be a programba, ami SQL kifejezésen alapul. Arra használjuk, hogy SQL lekérdezéseket hajtsunk végre a segítségével. Azért hasznos ez a komponens, mert ezzel egyszerre több táblát is



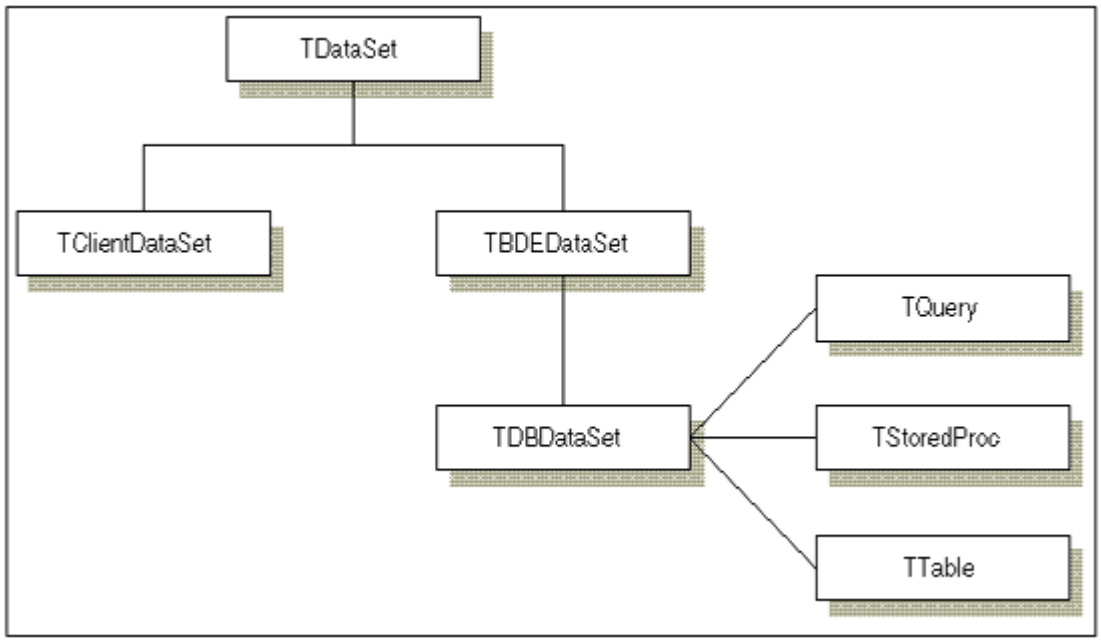
6. ábra

elérhetünk, illetve összekapcsolhatunk, valamint számításokat, rendezéseket, leválogatásokat is nagyon könnyen végezhetünk vele az SQL nyelvből adódóan. A Query komponens ugyanúgy rendelkezik a DatabaseName tulajdonsággal, mint a Table, de nincs TableName tulajdonsága. A táblát mindig az SQL utasításban adjuk meg, melyet az SQL tulajdonság tárol.

Az SQL utasításokkal egyszerre egy vagy több adattábla is elérhető a Query vezérlőelem DatabaseName tulajdonsága értékeként megadott adatbázisból. Az utasításokat az SQL tulajdonságában kell megadni. Ezt követően engedélyezni kell a lekérdezést, amit a kódban általában az Open() metódus hívásával, fejlesztés alatt pedig az Active tulajdonság true-ra való állításával tehetjük meg.

A lekérdezések végrehajtási teljesítményének növelése érdekében az első lekérdezést megelőzően érdemes meghívni a Query vezérlőelem Prepare() metódusát. A Prepare() metódus a BDE és az adatbázis-kiszolgáló inicializálását végzi, és rendszer erőforrásokat foglal le a lekérdezés számára. Ha az ExecSQL() metódus hívása előtt nem hívjuk meg a Prepare() metódust, az előkészítő folyamat e nélkül is lefut a lekérdezés végrehajtását megelőzően.

Az adatelérési komponensek a TDataSet leszármazottai.



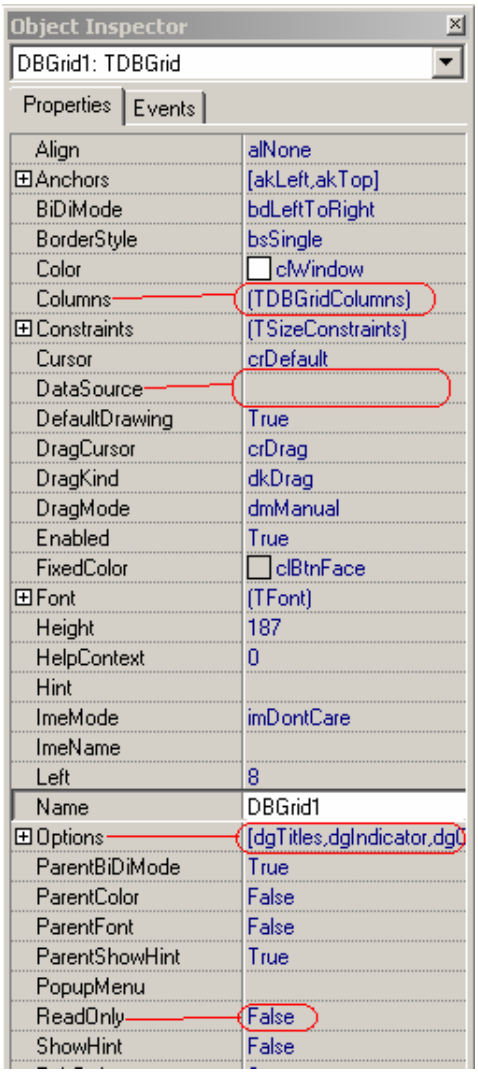
7. ábra

6.2. Data Controls komponensek

Ide az adatokat megjelenítő, kontrolláló vizuális komponensek tartoznak. Csupán megjelenítési célokra kifejlesztett komponensek, melyek az adathozzáférést biztosító komponensekhez kapcsolódnak, így ezek pillanatnyi állapotát tükrözik.

- ***DBGrid komponens***

A DBGrid segítségével táblázatos formában jeleníthetőek meg egy adatforrás rekordjai. Egyszerű táblamegjelenítő komponens. Egy TTable vagy TQuery objektum tartalmát jeleníti meg standard formában. Csak a DataSource property-t kell beállítani, és már láthatóvá is válnak az adott adatforrás által

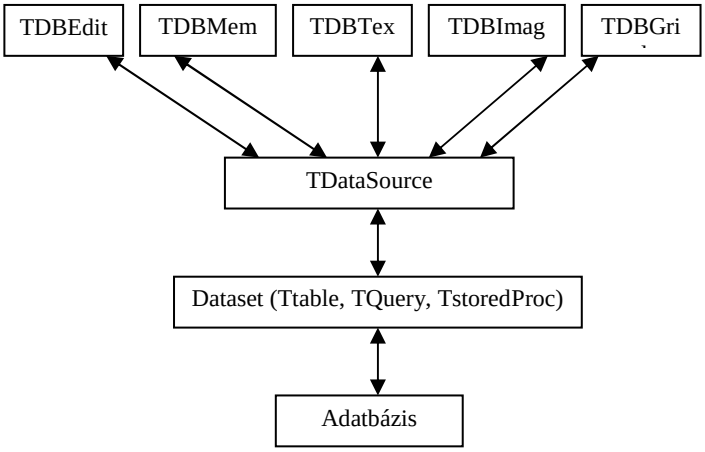


8. ábra

szolgáltatott adatok. A tábla tartalmát szerkeszteni is tudjuk a segítségével, amennyiben a ReadOnly tulajdonsága false. A szerkesztés a standard táblázatkezelő programokhoz hasonlóan történik. A komponens Options tulajdonságát kinyitva rengeteg beállítási lehetőséget találunk. A dgEditing tulajdonságot false-ra állítva kikapcsolhatjuk a szerkesztő módot, így az adatokat nem lehet módosítani. A dgRowSelect bekapcsolásával egyrészt automatikusan kikapcsoljuk a szerkesztést, másrészt engedélyezzük, hogy a táblázatnak mindig egy teljes sora legyen kijelölve. Sokszor lehet arra szükség, hogy a táblázatból csak meghatározott mezőket jelenítsünk meg, illetve hogy csak meghatározottakat tudjunk szerkeszteni. Erre szolgál a DBGrid oszlop-szerkesztője (Column Editor), amit a Columns tulajdonság gombjára kattintva érhetünk el, ezen kívül megnyithatjuk ezt az ablakot a komponensre való dupla kattintással, illetve a helyi menüjéből is. Alapértelmezésben ez az ablak üres, tehát a táblázat minden mezőt megjelenít, amit a hozzá tartozó Table tartalmaz.

- ***DBNavigator komponens***

Ez egy egyszerű komponens, mely csupán az alapvető tábla-navigációs parancsok kiadására szolgál. A DBNavigator komponens vezérlőgombokat tartalmaz, mellyel a felhasználó egy adathalmaz rekordjain lépkedhet az első vagy az utolsó rekordra, új rekordot vihet fel, illetve meglévőt törölhet. Természetesen a DBNavigator gomb esetében is az első, amit be kell állítanunk, az a DataSource property. A Hints property segítségével egy sztringlistában minden egyes gombra beállíthatjuk, hogy mi jelenjen meg útmutatásként, ha az egeret az adott gomb fölé visszük. Ahhoz, hogy a hintek megjelenjenek, a Showhint property-t true-ra kell állítani. Ha a ConfirmDelete property igaz, akkor a törlés gombot megnyomva egy dialógusablak jelenik meg, ahol meg kell, hogy erősítsük törlési szándékunkat. A VisibleButtons property segítségével azokat a gombokat el tudjuk tüntetni, amelyekre nincs szükségünk az adott feladathoz.



9. ábra

7. JELENTÉSKÉSZÍTŐ - *QUICKREPORT*

Az adatbázis-kezeléssel egyidős az az igény, hogy egy adatfeldolgozási folyamat eredménye jól áttekinthető formában papíron, vagy esetleg FAX-on elküldhető formában is megjelenjen. Az adatbázis-kezelő alkalmazások papírra irányuló kimenetét jelentésnek hívjuk. A Delphi rendszerben a QReport komponenslap komponenseit használjuk jelentések összeállításához. A QucikReport segítségével adatainkról az egyszerű listáktól kezdve a bonyolult, pl. diagrammokat, különböző kimutatásokat is tartalmazó jelentésig számos nyomtatvány elkészítésére nyílik lehetőségünk.

Az egész QuickReport alapját a QuickRep komponens képzi. Amikor egy jelentést készítünk, mindig először egy QuickRep komponenst kell a form-ra helyeznünk, majd erre tesszük a többi komponenst, melyek a jelentésünket meg fogják jeleníteni. Egy jelentés különböző szakaszokból (pl. fejléc, törzs, lábléc stb.) épül fel, és ezeken a szakaszokon helyezzük el azokat a komponenseket, amelyek az adatokat, szövegeket, képeket ténylegesen megjelenítik. A szakaszokat a QRBand komponens reprezentálja.

7.1. A QRBand komponenes

A QRBand komponens segítségével állíthatjuk be a jelentés különböző szakaszait. Azt, hogy a komponens éppen milyen szakaszt reprezentál, a BandType property segítségével lehet beállítani.

- **Jelentésfejléc**

  Egyszerre jelenik meg a jelentés legelső oldalán. Ebben szoktuk feltüntetni a jelentés címét.

- **Jelentéslábléc**

  A jelentés utolsó oldalán jelenik csak meg, benne végösszegzéseket szoktunk megjeleníteni.

- **Jelentéstörzs**

  Az ide helyezett információk a forrás-adathalmaz minden egyes rekordjára megismétlődnek.

A gyakrabban használt szakaszok a QuickReportban:

Band Type	**Megnevezés**
RbColumnHeader	Oszlop fejléc
RbDetail	Jelentéstörzs
RbGroupFooter	Csoport Lábléc
RbGoroupHeader	Csoport Fejléc
RbPageFooter	Oldal Lábléc
RbPageHeader	Oldal Fejléc
RbSubDetail	Master-Detail kapcsolatban a masterhez kapcsolódó adatok
bSummary	Szumma
RbTitle	Cím

Ha csoportosításokat is végzünk a jelentésünkben, akkor a csoportnak lehet egy fejléce és egy lábléce. Ezen kívül a jelentésben létrehozhatunk még oldalfejléc és lábléc szakaszokat is. Itt leginkább általános információkat szoktunk megjeleníteni, mint például az oldalszámot, a jelentést készítő cég emblémáját, a jelentés címét. Jelentésünk tartalmazhat még oszlopfejléc szakaszt is, ez minden oldalon megjelenik a lista oszlopainak fejlécével.

7.2. Megjelenítési Komponenesek

A megjelenítés komponenseket három csoportba oszthatjuk. Az első csoport olyan statikus adatok megjelenítésére használható, mint pl. egy szöveg, egy kép vagy egy keret, vonal stb. Ezek a komponensek nagyon hasonlítanak a standard VCL megfelelőikhez. A következő komponensek tartoznak ebbe a csoportba: QRLabel, QRShape, QRImage, QRRichText, QRMemo, QRExprMemo.

A második csoportba a VCL adatmegjelenítési komponensek QuickReport-os megfelelői tartoznak. Ezek segítségével egy adathalmaz adatait jeleníthetjük meg jelentéseinkben. A következő komponensek tartoznak ebbe a csoportba: QRDBText, QRDBRitchText, QRDBImage. Az első kettő segítségével az adatbázisban tárolt szövegeket, számokat, míg az utolsóval BLOB típusú mezőben tárolt képeket jeleníthetünk meg.

A harmadik csoportba két komponens tartozik, ezek a QRExpr és a QRSysData. A QRExpr komponenssel a QuickReport beépített függvényeit használhatjuk fel és jeleníthetjük meg. A QRExpr komponenst a Summary Band-en helyezzük el, majd az Expression property-ben adjuk meg a megfelelő kifejezést (pl. COUNT()-ra összeszámolja a törzsszakaszban megjelenített sorok számát). Ha az Object Inspectorban az Expression tulajdonság gombjára kattintunk, akkor bejön a kifejezés-varázsló, ahol könnyedén összeállíthatjuk a megfelelő kifejezést, és a Validate gombot megnyomva a QuickReport még le is ellenőrzi nekünk, hogy szintaktikailag helyes kifejezést adtunk-e meg. A QRSysData komponenssel a jelentés készítésének dátumát, idejét, az adott oldal számát stb. jeleníthetjük meg. Hogy mi is jelenjen meg, azt a Data proprty-ben tudjuk beállítani.

7.3. QuickReport Master/Detail

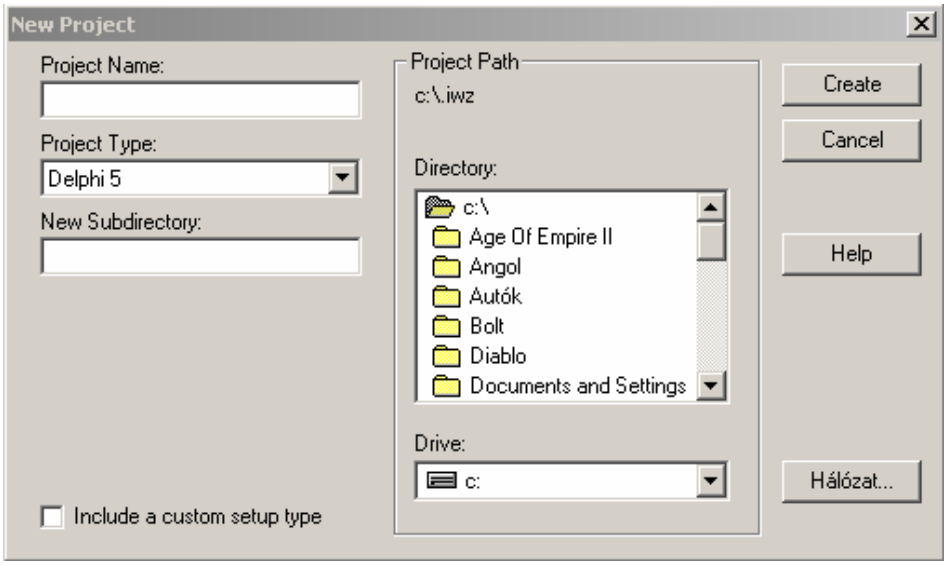
A „QuickReport Master/Detail” sablon segítségével olyan jelentéseket készíthetünk, amelyekben egy, az ún. alaptáblában (master table) található valamelyik mező értéke alapján kikeressük az összes megfelelő rekordot egy másik, ún. részletező táblában (detail table).

Ebben az esetben a beszámoló készítéséhez két adatkészletező (TTable) és egy adatforrás vezérlőre (TDataSource) van szükség. Az adatkészletező komponenseknél beállítjuk a DatabaseName, valamint TableName tulajdonságokat, és az adatforrás vezérlő DataSet tulajdonságát az alaptáblára állítjuk. Megadjuk a két tábla közötti kapcsolatot, a részletező tábla MasterSource tulajdonságában beállítjuk az alaptáblára mutató adatforrás vezérlőt és a MasterFields tulajdonságánál megadjuk, hogy melyik két mező alapján kötjük össze a táblákat. Ha beállítottuk a jelentésünk formai elrendezését, tehát megadtuk az oldalfejlécet (Page Header), oszlopfejlécet (Column Header), akkor szükségünk lesz az alaptábla adatainak a megjelenítésére, amit a jelentéstörzsbe (Detail) helyezhetünk el. Ezek után nincs más teendőnk, mint az alaptábla megfelelő soraihoz tartozó részletező tábla sorainak megjelenítése. Ezt a QRSubDetail szakaszban tehetjük meg a következőképpen. Meg kell adnunk a DataSet tulajdonságában, hogy ennek a szakasznak melyik az alaptáblája. Beállítjuk, hogy mely mezőket szeretnénk megjeleníteni. Ha a jelentéstörzs (Detail) ForceNewPage tulajdonságát true-ra állítjuk, akkor elérhetjük vele, hogy az alaptábla sorai mindig új lapon kezdődjenek. A 1. számú mellékletben látható egy demo adatbázis nyomtatási képe.

8. TELEPÍTŐKÉSZLET ELŐÁLLÍTÁSA (*INSTALLSHIELD EXPRESS)*

Az InstallShield Express alkalmazás szűkített (Limited Edition), a Delphi környezet sajátosságaira specializált ingyenes változatát megtalálhatjuk a Delphi telepítő lemezén. Segítségével a Delphi rendszerben írt alkalmazásokhoz telepítő-készletet készíthetünk a különböző Windows platformok alá.

A kész programunk telepítőkészletének elkészítéséhez az InstallShield Express alkalmazás elindítása után ki kell választanunk az új projekt készítését (Create a New Setup Project), és a megjelenő New Project párbeszédablakban meg kell adnunk a telepítőkészletre vonatkozó név- és útvonal-információkat.



10. ábra

A projekt adatainak beállítása után az InstallShiled Express ablakában megjelenik a feladatlista (Setup Checklist), amelyben egymás után rá kell kattintanunk a feladatkiírások mellett szereplő nyilakra, hogy egymás után elvégezzük a telepítő készlet előállításához szükséges lépéseket. A beállítások után a feladatlista megfelelő bejegyzése előtt egy piros színű pipa jelenik meg. A nyilakra való kattintás helyett a Checklist menüből is kiválaszthatjuk a szükséges lépéseket.

A feladatlista első csoportja (Set the Visual Design) a készülő telepítő készlet általános adatainak beállítását célozza, illetve a telepítési folyamat hátterének megtervezését segíti. A csoport bármelyik bejegyzése előtt álló nyílra kattintva a „Set the Visual Design” párbeszédablak jelenik meg. A párbeszédablak első (App Info) lapján annak az alkalmazásnak a nevét, elérési útját és verziószámát kell megadnunk, amelynek a telepítőkészletén dolgozunk. Az ablak Company és a „Default Destination Directory” mezőjében adhatjuk meg az alkalmazást fejlesztő cég nevét, illetve a

telepített alkalmazás könyvtárát. A párbeszédablak második lapján (Main Window) a telepítő ablakának hátterére vonatkozó adatokat adhatjuk meg. A „Main Title” mezőben definiálhatjuk a megjelenő feliratot, illetve képet. A „Logo Bitmap” keretben jelvényünket és annak ablakon belüli pozícióját (Position), illetve az ablak háttérszínét (Background Color) állíthatjuk be. Ahhoz pedig, hogy a telepítőkészletünk a telepített alkalmazást leszedő (uninstall) programot is tartalmazza, be kell jelölnünk az „Automatic Uninstaller” jelölőnégyzetet, a „Set the Visual Design” párbeszédablak utolsó, (Features) lapján.

A feladatlista „Specify InstallShield Objects for Delphi” bejegyzésének segítségével az alkalmazásunk futásához szükséges Delphi objektumokat, illetve objektumcsomagokat adhatjuk meg a megjelenő párbeszédablak General lapján. Az InstallShield Expres létrehozza a System Files – WinSysDir nevű fájlcsoportokat, és elvégzi a szükséges állomány-, registry- és rendszerfájl-módosításokat.

A „Specify Components and Files” párbeszédablakban a telepítendő állományokat csoportokba (Groups) szervezhetjük. A kialakított fájlcsoportokat az alkalmazás összetevőihez (Components) sorolhatjuk. Az alkalmazás összetevőit a telepítőkészlet típusának (Setup Types) kiválasztásakor használjuk. Alaphelyzetben teljes (Complete) telepítést lehetővé tevő készletet állíthatunk elő. Amennyiben a következő lépéshez tartozó „Dialog Boxes” párbeszédablakban bejelöljük a „Setup Types” elemet, a lehetséges telepítési módok megváltoznak:

Telepítési mód **Leírás**

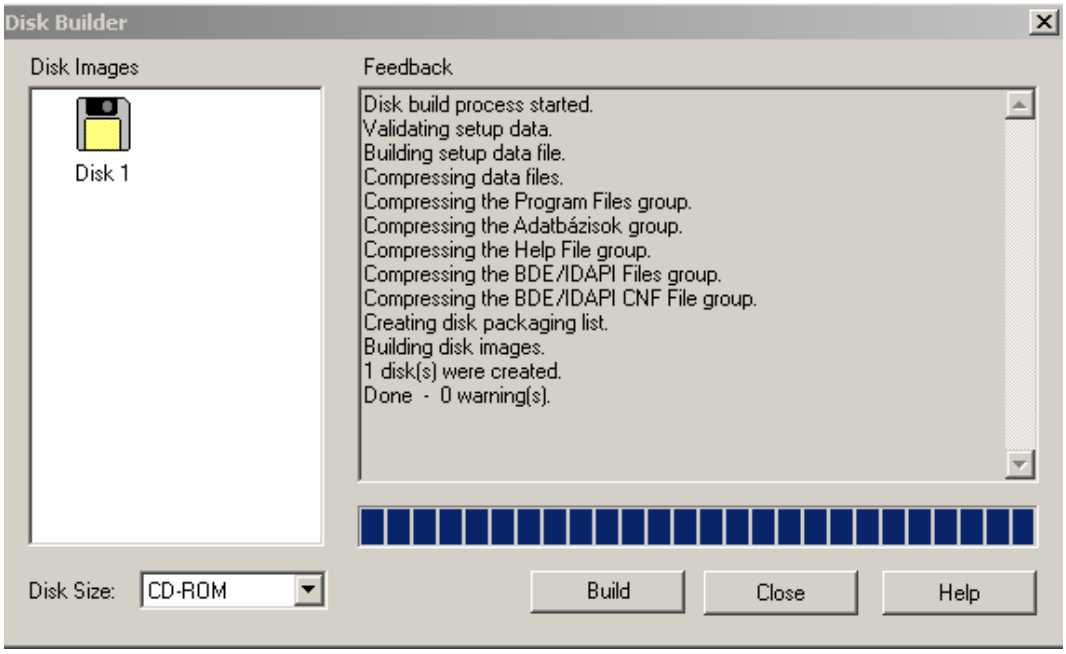
- Typical Általában az összes összetevőt magában foglaló teljes telepítést jelenti.
- Compact Általában az alkalmazás futtatásához feltétlenül szükséges összetevők telepítését jelenti.
- Custom A felhasználó választhat a telepítendő összetevők közül.

A „Specify Components and Files” párbeszédablak első, Group lapján telepítő-készletünket új fájlcsoportokkal bővíthetjük (New Group). A fájlcsoportokhoz hozzáadhatjuk az alkalmazás által használt állományokat (Insert Files). A csoportok jellemzőit (Properties) meg is változtathatjuk a „Modify Group” párbeszédablakban. A fentiekhez hasonló párbeszédablakokban a fájlcsoportokból kialakíthatjuk az alkalmazásunk összetevőit, illetve megválaszthatjuk a telepítés típusát.

A következő feladatcsoport, a „Select User Interface Components” párbeszédablak a telepítési folyamat jellemzőinek beállítását hivatott segíteni. A „Settings For” listában bejelölt tulajdonságok egy részét – mint például az első párbeszédablakban megjelenő kép (Welcome Bitmap), vagy a felhasználói információk (User Information) – a Settings lapon megváltoztathatjuk.

A „Specify folders and Icons” feladatcsoportot nyilain kattintva, a megjelenő párbeszédablak General és Advanced lapjain alkalmazásunk indítására, illetve futtatására vonatkozó beállításokat adhatjuk meg, például a parancssor paramétereit (Run Command Parameters), az alkalmazás mappáját (Folder), ikonját (Icon) stb.

Ha minden adatot megadtunk, akkor a „Run Disk Bulider” feladatcsoportból elindíthatjuk a telepítőkészlet létrehozását. A párbeszédablak jobb oldali panelján az



11. ábra

összeállítási folyamat kísérő információit láthatjuk, például a figyelmeztetéseket (Warning) is. (11. ábra)

Ha az összeállítási folyamat sikerrel járt, elindíthatjuk a létrejött telepítő-készletünk tesztelését a „Test the Installation” feladatcsoportból. A telepítési folyamat ugyanúgy megy végbe a számítógépünkön, mint később a felhasználó rendszerében.

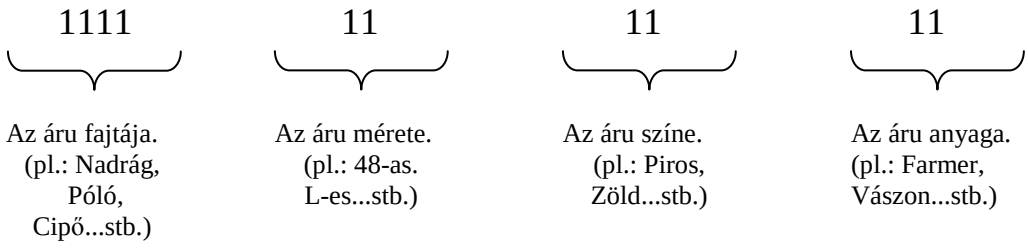
Ha mindezek után létre szeretnénk hozni az alkalmazás telepítőlemezét is, a „Create Distribution Media” feladatcsoport „Copy to Floppy” ikonján kell kattintanunk.

9. A KÉSZLETNYILVÁNTARTÓ PROGRAM BEMUTATÁSA

Mivel a szakdolgozati témámban szereplő vállalkozás még kicsi, ezért a nagy terjedelmű, bolti hálózatokra kifejlesztett szoftver megvásárlása túlságosan költséges lenne, valamint nem minden funkcióját lehetne maradéktalanul kihasználni. Ráadásul előtérbe került, hogy egy saját, egyedi kódolással lenne célszerű a nyilvántartást vezetni. A rendszert fel kell készíteni arra, hogy a termékek betárolása, illetve kiadása vonalkód-leolvasó berendezéssel fog történni. Ezért nélkülözhetetlenné vált egy egyedi tervezésű rendszer megalkotása.

9.1. A vonalkód felépítése

A nyilvántartás alapja, a termékazonosító úgy lett megtervezve, hogy a cikkek különböző tulajdonságaik alapján kapnak csak más azonosítót. Ez azt jelenti, hogy a piros farmernadrág, melynek mérete 48-as, külön kódot kap a zöld farmernadrágtól, ami szintén 48-as méretű. A kód egy 10 karakter hosszú számjegy, amely a következő-képpen épül fel.



Ebből is látszik, hogy 4 tábla máris szükséges, hogy ezeket a kódokat tárolni és karbantartani tudjuk. A táblák struktúrája a következő:

Áru.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
KOD	NUMERIKUS	4	Az áruhoz tartozó kód.
NEV	KARAKTERES	50	Az áru megnevezése.

Indexállomány: ARU.MDX **Indexkulcs:** KOD

Méret.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
KOD	NUMERIKUS	2	A mérethez tartozó kód.
NEV	KARAKTERES	50	A méret megnevezése.

Indexállomány: MERET.MDX **Indexkulcs:** KOD

Szín.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
KOD	NUMERIKUS	2	A színhez tartozó kód.
NEV	KARAKTERES	50	A szín megnevezése.

Indexállomány: SZIN.MDX **Indexkulcs:** KOD

Anyag.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
KOD	NUMERIKUS	2	Az anyaghoz tartozó kód.
NEV	KARAKTERES	50	Az anyag megnevezése.

Indexállomány: ANYAG.MDX **Indexkulcs:** KOD

A fenti négy tábla felhasználásával megalkotjuk a vonalkódjainkat, melyet egy árutörzs nevű táblában tárolunk. Itt már a ténylegesen összeszerkesztett azonosító van elhelyezve a megfelelő árumegnevezéssel, valamint egy mennyiségi egységgel.

Arutorzs.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
KOD	NUMERIKUS	10	A cikk vonalkódja.
NEV	KARAKTERES	50	A cikk megnevezése.
MENYEGY	KARAKTERES	30	A cikk mennyiségi egysége.

Indexállomány: ARUTORZS.MDX **Indexkulcs:** KOD

9.2. A termékek nyilvántartása

A feladat szerint az áruinkat nyilván kell tartani, tehát fel kell vennünk a készletbe. Ez nem jelent mást, mint az árutörzs-adatbázis alkalmazásával a készlet-táblánk feltöltését.

Keszlet.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
KOD	NUMERIKUS	10	Az áru vonalkódja.
BEAR	NUMERIKUS	8	Beszerzési ára.
HASZON	NUMERIKUS	2	Haszonkulcs százalékban.
AFA	NUMERIKUS	2	Az Áfa mértéke.
KESZDAT	DÁTUM		Az utolsó készletbevétel dátuma.
DARAB	NUMERIKUS	5	Készleten lévő darabszám.

Indexállomány: KESZLET.MDX **Indexkulcs:** KOD, KESZDAT

Ha a készlet-adatbázis még nem tartalmaz olyan vonalkódot, amit készletbe veszünk, akkor új rekordként felvitelre kerül, más esetben, tehát ha már tartalmazza a felvinni kívánt kódot, akkor a beszerzési ár, a haszonkulcs, az Áfa mértéke és a Készletbevétel dátuma felülíródik, valamint a készleten lévő darabszámot növeljük az éppen felvinni kívánt darabszámmal.

A termékeket nem csak készletbe kell vennünk, hanem eladnunk is, ami nem jelent mást a rendszer szempontjából, mint a készlet csökkentését. Ez úgy működik, hogy az eladáskor az éppen aktuális tételeinket összegyűjtjük egy ideiglenes táblába,

melynek a neve eladás. Rendkívül szükséges, mivel egy áramszünet esetén elveszhetnek a tételeink. A tényleges eladást követően csökkentjük a készlet táblát az eladás adatbázis adataival, valamint az eladás adatbázis teljes tartalmát átmozgatjuk az eladva állományba. Erre azért van szükség, hogy a már eladott termékeket a későbbiek folyamán vissza tudjuk keresni.

Eladas.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
KOD	NUMERIKUS	10	Az áru vonalkódja.
NEV	KARAKTERES	50	Az áru megnevezése.
DARAB	NUMERIKUS	5	Eladott darabszám.
ELADAR	NUMERIKUS	5	Eladási ár.

Indexállomány: NINCS **Indexkulcs:** NINCS

Eladva.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
ELADAT	DÁTUM		Eladás dátuma.
SORSZ	NUMERIKUS	4	Vevő sorszáma
KOD	NUMERIKUS	10	Az áru vonalkódja.
DARAB	NUMERIKUS	5	Eladott darabszám.
ELADASIAR	NUMERIKUS	5	Eladási ár.

Indexállomány: ELADVA.MDX **Indexkulcs:** ELADAT, KOD

Az eladás folyamán a végösszegből bizonyos kedvezményt is kaphat a vevő, ezért fontos, hogy tudjuk, melyik nap hányadik vevőhöz párosítsuk ezt a paramétert.

Sorszam.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
SORSZAM	NUMERIKUS	4	Vevő sorszám.
DATUM	DÁTUM		Az aktuális dátum.

Indexállomány: NINCS **Indexkulcs:** NINCS

Kedvez.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
DATUM	DÁTUM		Vásárlás dátuma.
SORSZ	NUMERIKUS	4	Vevő sorszáma aznap.
KEDV	NUMERIKUS	2	Kedvezmény mértéke százalékban.

Indexállomány: NINCS **Indexkulcs:** NINCS

Ennyi tábla elég is lenne a termékek készletnyilvántartásához, de sok esetben szeretnénk nyomtatott formában is képet kapnunk a keresési feltételünknek eleget tévő rekordokról, így 2 darab átmeneti tábla is tartozik a rendszerhez. Erre azért van szükség, hogy egyszerűbb legyen az adatok összegyűjtése. A feltöltött tábla rekordjait csak sorban ki kell így nyomtatni.

Templist.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
ELADAT	DÁTUM		Eladás dátuma.
SORSZ	NUMERIKUS	4	Vevő sorszáma.
KOD	NUMERIKUS	10	Az áru vonalkódja.
NEV	KARAKTERES	50	Az áru megnevezése.
DARAB	NUMERIKUS	5	Eladott darabszám.
EREDETIAR	NUMERIKUS	8	Eredeti eladási ár.
KEDVAR	NUMERIKUS	5	Kedvezményes eladási ár.
KEDV	NUMERIKUS	2	Kedvezmény.
FIZAR	NUMERIKUS	5	Fizetendő ár.

Indexállomány: TEMPLIST.MDX **Indexkulcs:** ELADAT, KOD

Templis1.dbf állomány felépítése

Mezőneve	Típusa	Hossza	Jelentése
KESZDAT	DÁTUM		Készletbevétel dátuma.
KOD	NUMERIKUS	10	Az áru vonalkódja.
NEV	KARAKTERES	50	Az áru megnevezése.
DARAB	NUMERIKUS	5	Készleten lévő darabszám.
BEAR	NUMERIKUS	8	Beszerzési ár.
HASZON	NUMERIKUS	2	Haszonkulcs mértéke százalékban.
AFA	NUMERIKUS	2	Az Áfa mértéke.
ELADASIAR	NUMERIKUS	5	Eladási ár.

Indexállomány: TEMPLIS1.MDX **Indexkulcs:** KESZDAT, KOD

Tehát mint láthatjuk, nem minden táblát kell a felhasználónak karbantartania, hisz vannak olyanok is, amelyeket a rendszer saját maga kezel.

10.A PROGRAM ÜZEMBE HELYEZÉSE

A szoftver használatba helyezésének az első és legfontosabb lépése, hogy rendszergazdai jogokkal legyünk bejelentkezve a számítógépünkre, valamint telepítsük a programot a számítógépünkre. Ez roppant egyszerű módon történik, hisz a szakdolgozathoz csatolt telepítő CD-ROM-on található SETUP.EXE állomány lefuttatásával a program és a hozzá kapcsolódó fájlok automatikusan felmásolódnak a gépünkre, továbbá beállításra kerül a BDE Administrator szoftverben az alias név is. Erre azért van szükség, hogy a program által használt táblákat kezelni tudja rendszer.

A szoftver csak 32 bites Windows alapú operációs rendszereken fut. Ettől eltérő rendszerre nem lehet telepíteni, tehát a minimális hardver igény megfelel a Microsoft által támasztottaknak. Ez a következő:

- 486DX, 66 MHz-es vagy ennél nagyobb órajel-frekvenciájú processzor.
- 24 MB RAM (Több memóriával jobb teljesítmény érhető el)
- kb. 30 MB szabad hely a HDD-n
- CD-ROM vagy DVD-ROM meghajtó
- VGA vagy ennél nagyobb felbontású monitor
- Microsoft Mouse vagy ezzel kompatibilis egér
- Billentyűzet
- Nyomtató

10.1. Telepítési folyamat bemutatása

A telepítés elindításakor egy üdvözlő kép tárul elénk. A folyamat folytatásához nyomjuk meg a next gombot, és máris egy beállítási lehetőség kínálkozik fel. Ez nem más, mint a start menü programok listája funkcióban megjelenő mappa megnevezése. Alapbeállításként a következő szerepel: Bolti Nyilvántartás. Ha nem ide szeretnénk elhelyezni az indító parancsfájlt, akkor a megjelenő listából válasszuk a számunkra megfelelőt. A folytatás szintén a next gomb megnyomásával történik, és ezzel elindul az állományok felmásolása, valamint a szükséges paraméterek beállítása. Sikeres telepítés esetén már csak a befejező kép kínálkozik elénk. A finish gomb megnyomásával végzünk az installálással. A telepítőprogram az asztalunkra kihelyezi a programindító ikont.

12. ábra

Ha bármi oknál fogva nem található meg, akkor még másféleképpen is elindíthatjuk a készlet-nyilvántartó szoftvert.

- Start menü, futtatás funkciójánál a megnyitás sorba írjuk be a következőt: C:\Bolt\Keszlet.exe. majd nyomjuk meg az OK gombot.

10.2. Szükséges beállítások

A szoftver indítása előtt a következő beállítások elvégzése fontos a Windows operációs rendszerben a szoftver hibátlan működése szempontjából.

1. A Start menü – Beállítások funkció – Vezérlőpult ablakban a képernyő ikonra kattintva válaszuk ki a beállítások lapfület. A beállítások fülön található képernyőfelbontás csúszkát állítsuk addig, amíg alatta meg nem jelenik a következő felirat: 800 x 600 képpont. Ez a minimum képernyő-felbontás, aminél minden jól látszik. Ettől nagyobb lehet a felbontás. Ezek után az OK gomb lenyomásával fogadjuk el a változtatásokat.

2. A Start menü – Beállítások funkció – Vezérlőpult ablakban a területi beállítások ikonra kattintva válaszuk ki a dátum lapfület és ellenőrizzük a következő beállításokat:

    - Rövid dátumformátum: éééé.HH.nn.
    - Hosszú dátumformátum: éééé. HHHH nn.

    Az Ok gomb megnyomásával fogadhatjuk el a módosításainkat.

Az eredményes installálást és beállítást követően az alkalmazásom menüpontjaira térnék ki részletesebben.

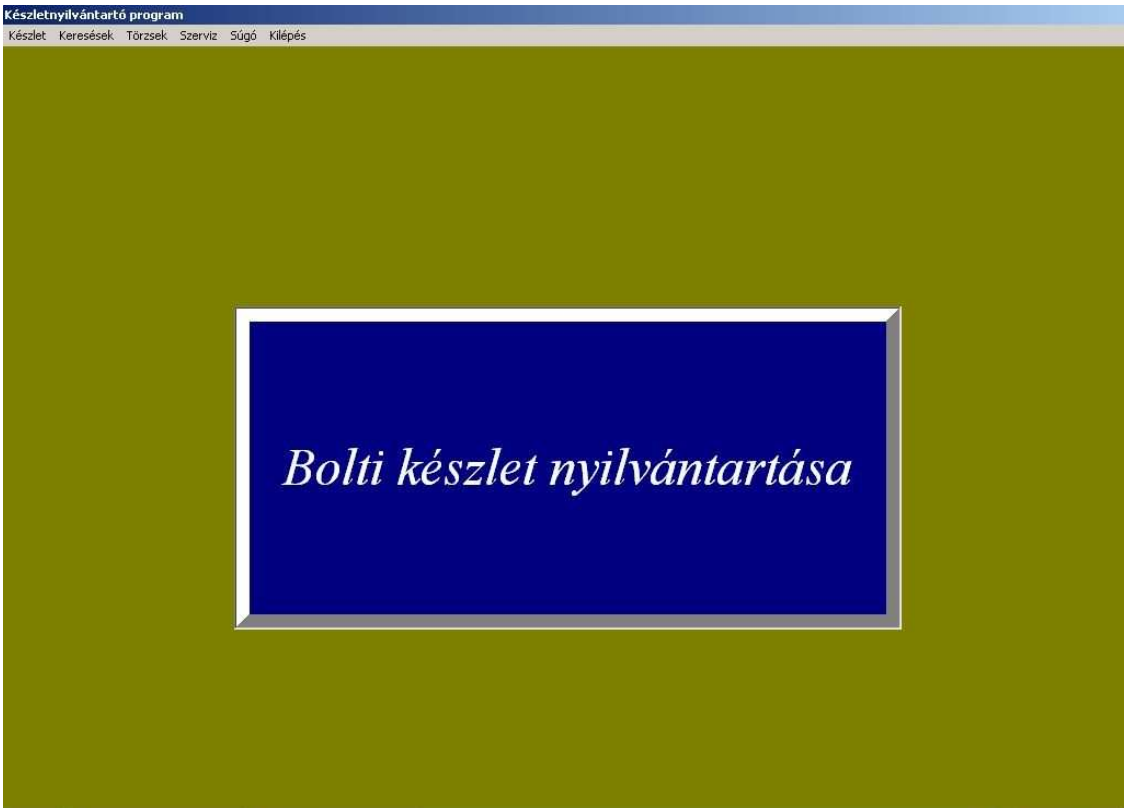
11.RÉSZLETES PROGRAMLEÍRÁS

A szoftver képes egyéni vonalkód szerkesztésére, melyet a törzsek menüpont alatt tudunk elvégezni, valamint egy külső vonalkód-nyomtató program segítségével etikett címkéket lehet készíteni, melyet ez a program tud kezelni. Tehát minden egyes árut készletbevételkor ellátunk egy vonalkóddal, majd az eladás során ezt a kódot felhasználva csökkentjük a készletet. A készletbevétel során kalkulálunk egy eladási

árat, melyet az árváltoztatás menüpont segítségével módunk van a későbbiekben megváltoztatni. A szoftver további lehetőségei, hogy a készletbe vett áruk között, valamint az eladott áruk között a megfelelő szempontok megadásával lehetőség van keresni, és a kapott eredményt nyomtatott formában is megkaphatjuk. Az adatbiztonságot figyelembe véve lehetőségünk van az adatállományainkról mentést készíteni, vagy egy régi mentést visszatölteni.

11.1. Főmenü

Itt választhatjuk ki, hogy a programon belül milyen funkciót szeretnénk elérni. A továbbiakban részletesen leírom, hogy mely menüpontok milyen célt is szolgálnak.



13. ábra

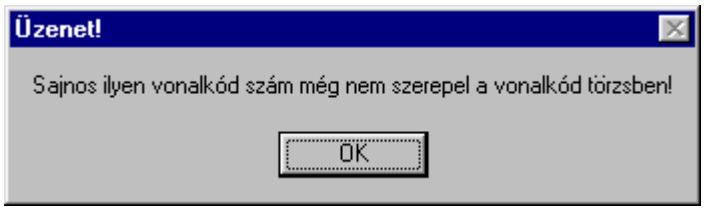
11.2. Készlet

Ezen menüpont alatt három modul található, melyből kettő a termékek készletbeli mozgásával, egy pedig a termék eladási árának a változtatásával foglalkozik.

11.2.1. Készletbevétel

Mint a menüpont elnevezése is utal rá, itt tudjuk az eladásra szánt árukészletünket feltölteni, bővíteni.

Az első és legfontosabb lépés a vonalkód törzsben ténylegesen létező vonalkód megadása. Rossz kód begépelése esetén üzenetet küld a program, mivel csak azokat a kódokat tudja értelmezni a rendszer, amely a saját törzsében szerepel.



14. ábra

Az OK gomb lenyomása után megismételhetjük a kód megadását. A helyes számsorozat megadását követően a program a kódhoz tartozó árumegnevezést és mértékegységet megkeresi és kiírja a képernyőre. A következő lépésként meg kell adnunk egy darabszámot. Ez azt jelenti, hogy ebből a fajta termékből ennyi darabot kívánunk a készletünkbe felvenni. Ezután az áruhoz tartozó mennyiségi egység jelenik meg, melyet a törzsből veszi, de lehetőségünk van itt módosítani. Ha mindezzel megvolnánk, akkor a tényleges beszerzési ár megadásával folytatjuk. Erre azért van szükség, mert a továbbiakban fogunk kalkulálni egy eladási árat. A készletbevétel dátumát kötelező megadni.

A rendszer csak jó dátumot fogad el, különben hibaüzenetet kapunk. Csak helyes dátum megadása után tudunk továbbhaladni.

15. ábra

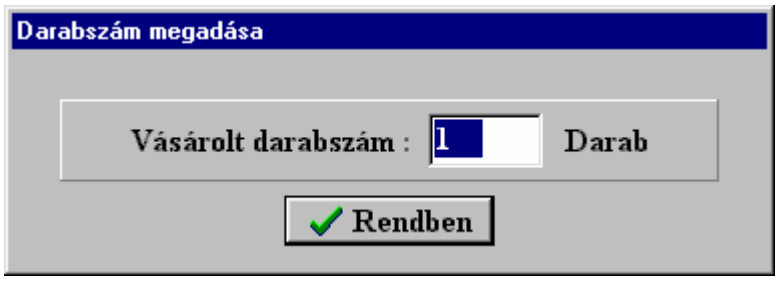
Az eladási ár kalkulációjához fontos paraméter a haszonkulcs. Ezt a következő lépés megtételével állíthatjuk be. Fontos megjegyezni, hogy az értéket százalékban kell megadni. A további lépésben az áfa kulcs kiválasztása lehetséges. Alapesetként a 20%-os áfa kulcs van beállítva. Ha a fenti mezőket sikeresen kitöltöttük, akkor az eladási ár kiszámítása gomb megnyomásával megkapjuk az egy termékre vonatkozó eladási árat. A képernyő alsó felében megtekinthetjük, hogy a cikk ára miként alakult. Ha nem megfelelő a számunkra, akkor lehetőségünk van a haszonkulcson módosítani, majd az eladási ár kiszámítása gomb újabb megnyomásával a módosított kalkulált árat kapjuk. Ha minden adat megfelelő, akkor nyomjuk meg a rögzítés gombot, és a készletbevétel ezzel megtörtént.

11.2.2. Eladás

Mivel a készletbe vett áruinkat szeretnénk értékesíteni is, ezért erre a funkcióra is nagy szükségünk van. Hasonlóképpen működik, mint a vonalkód olvasós pénztárgép. Leolvassuk, vagy begépeljük a terméken elhelyezett vonalkódot, és jöhet a következő termék.

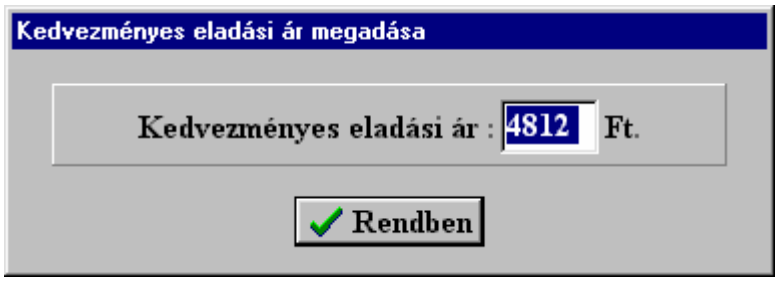
A helyes kód leolvasását követően megjelenik a termék neve és a kalkulált eladási ár. Rossz termékazonosító megadása esetén hibaüzenetet kapunk. Ha több azonos termékünk van, akkor a termékek vonalkódjának leolvasása előtt nyomjuk meg a darabszám megváltoztatása gombot, és adjuk meg, hogy hány darab azonos termékünk van, majd elég egy cikk vonalkódjának a leolvasása. Így egy vonalkód leolvasásával

ugyanazt érjük el, mintha az összes terméket egyesével leolvassuk. A szoftver figyeli, hogy a termékből van-e annyi darab a készletben, mint amennyi darabszámot megadtunk. Ha kevesebb van, akkor figyelmeztet a rendszer.



16. ábra

Lehetőségünk van mindig az utolsó leolvasott cikk eladási árának a megváltoztatására, ez a kedvezményes eladási ár megadása gomb segítségével történik.



17. ábra

A beírható legalacsonyabb összeg mindig a beszerzési ár, a beírható legmagasabb összeg pedig a kalkulált eladási ár lehet. Ha bármelyik határértéket átlépjük, akkor figyelmeztetést kapunk a rendszertől.

A középtájon elhelyezett táblázatban láthatjuk a már leolvasott termékek kódját, nevét, hány darab lett vásárolva és az eladási árat. Ha a darabszám egy, akkor az eladási árat láthatjuk, de ha nem egy, akkor a darabszám szorozva az eladási árral összeget olvashatjuk az utolsó oszlopban.

Szükség esetén lehetőség van arra, hogy töröljük a már leolvasott tételek közül bármelyiket. Kiválasztjuk a táblázatból a megfelelő sort, és megnyomjuk a kijelölt tétel

18. ábra

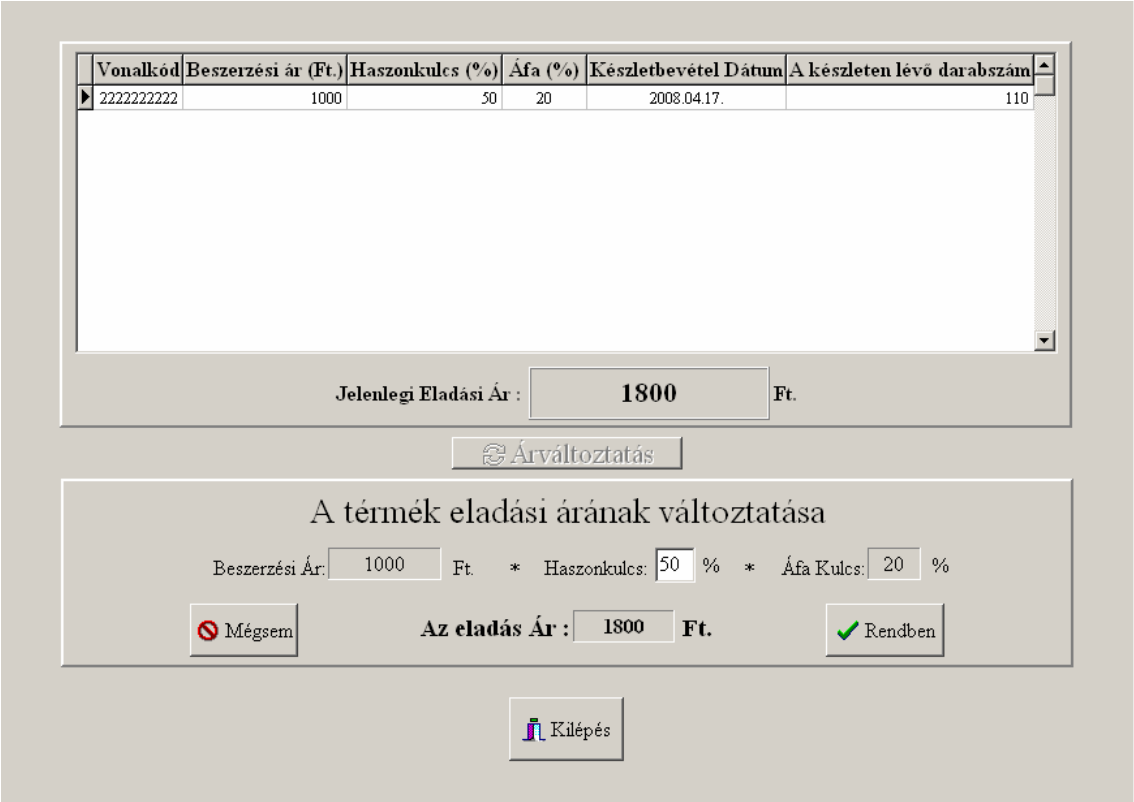
törlése billentyűt. Ha a vásárlónak az összes termékét leolvastuk, akkor adhatunk a fizetendő összegből kedvezményt. A beírt százalékérték után mindig nyomjunk egy ENTER billentyűt, hisz ennek a hatására láthatjuk a kedvezménnyel csökkentett fizetendő összeget.

Bármilyen okból kifolyólag, ha ki kell lépnünk az eladás gomb megnyomása előtt, akkor is minden adat megmarad, tehát egy áramszünet esetén is folytatható minden ott, ahol abbahagytuk. Az eladás gomb megnyomásával csökkentjük a készletünket az eladott termékekkel, és megkapjuk a fizetendő összeget, majd jöhet a következő vásárló.

11.2.3. Árváltoztatás

Mint a mindennapi élet is igazolja, akciók szervezésére szükség van. Bizonyos cikkek árából engedményeket kell adnunk a meglévő készlet csökkentése miatt. Tehát itt tudjuk a készleten lévő áruk eladási árát módosítani.

A listából válasszuk ki azt az árut, amelynek az árát meg szeretnénk változtatni. A lista alatt látható az aktuális kalkulált eladási ár. Az árváltoztatás gomb megnyomásakor lehetőségünk van a haszonkulcs módosítására. A beírt százalékérték után mindig nyomjunk egy ENTER billentyűt, hisz ennek a hatására láthatjuk az újból kiszámított eladási árat. Ha a mégsem gombot nyomjuk meg, akkor nem történik meg az árváltoztatás, viszont ha a rendben billentyűt kattintjuk le, akkor a már módosított haszonkulccsal kikalkulált új eladási ár kerül a nyilvántartásunkba.



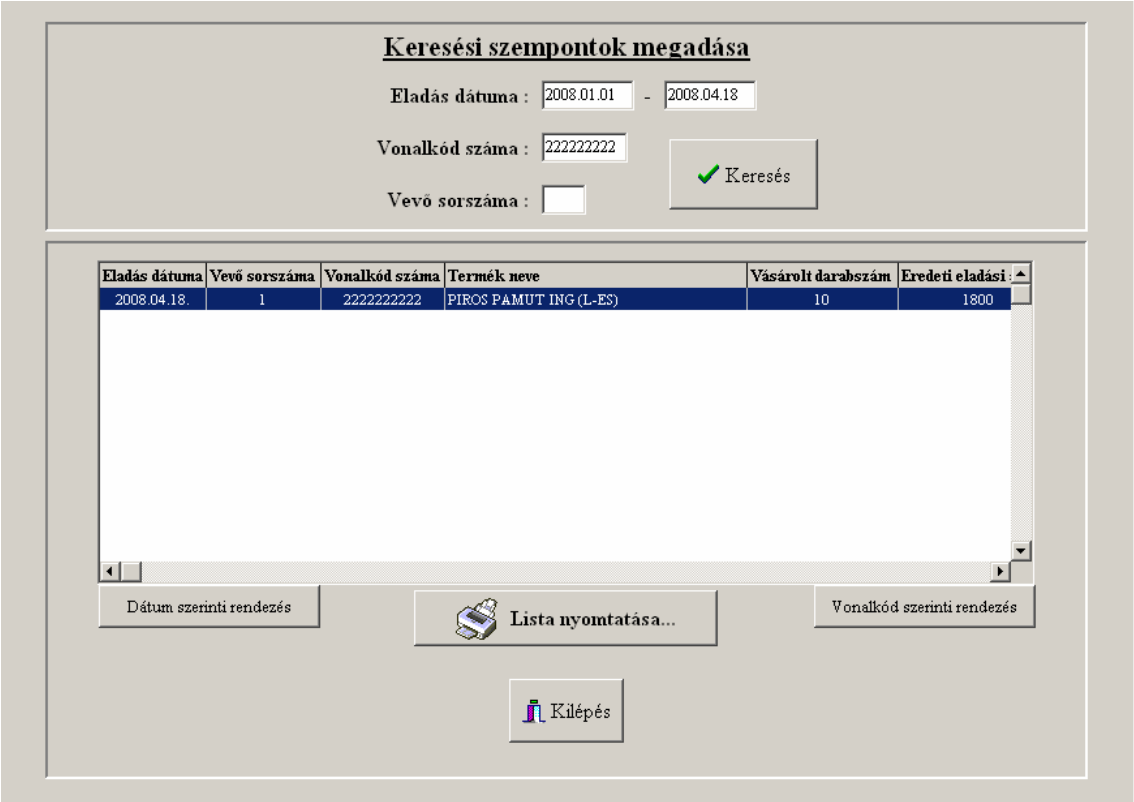
19. ábra

11.3. Keresések

Egy nyilvántartással foglalkozó szoftvernek mindig tudnia kell a különböző adatok között visszakeresni. Az itt található funkciók az eladott termékek és a készletben lévő cikkek között tud a megfelelő szempontok megadása mellett listát készíteni a képernyőre, illetve a nyomtatóra.

11.3.1. Eladott áruk közötti keresés

Az eladott áruk visszakeresése sokszor nagyon hasznos dolog. Ez a menüpont erre ad lehetőséget. Ez a funkció nagyon jól használható a napi zárás lista nyomtatására is. Három szempontot tudunk megadni, amelyből a listázási intervallum megadása a kötelező, a vonalkód számának, valamint a vevő sorszámának a megadása választható. Minél több szempontot adunk meg, annál jobban szűkítjük az eredményhalmazunkat.



20. ábra

Az intervallum megadásánál a helytelen dátum megadásakor hibaüzenetet küld a rendszer.

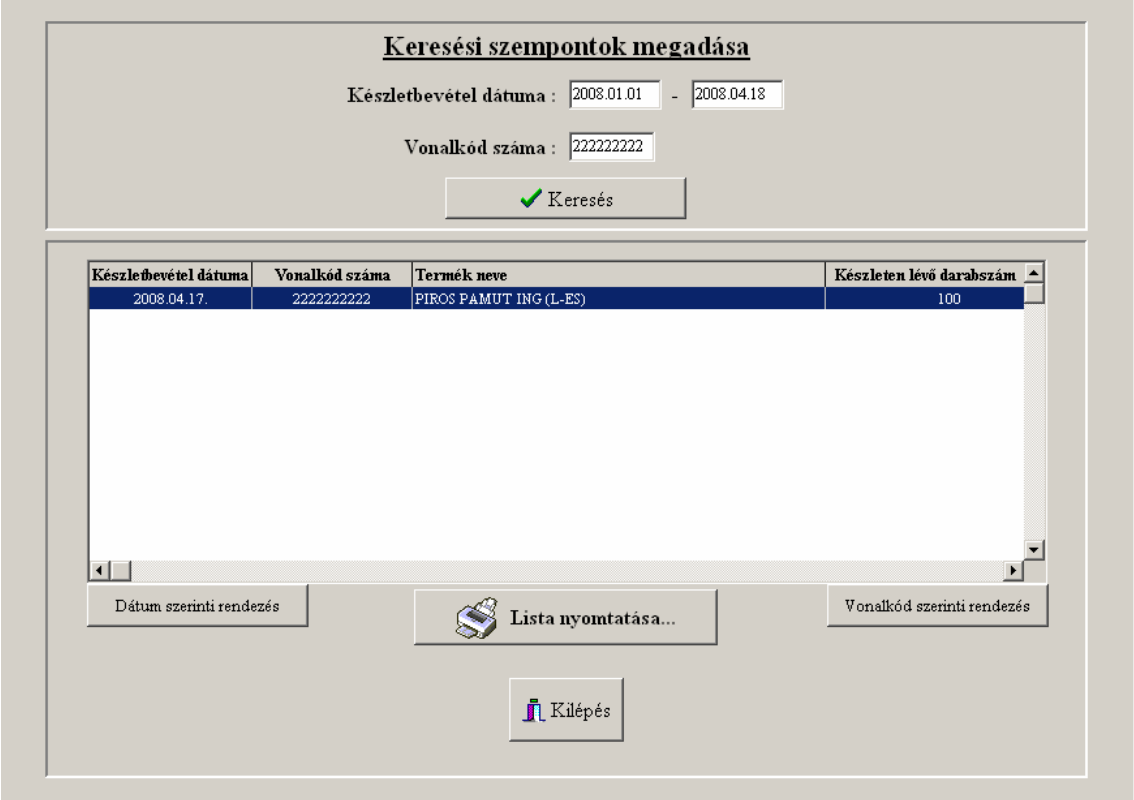
A keresés gomb megnyomásával indítjuk el a keresési folyamatot. A képernyőn megjelenő listában a gördítő sávok segítségével mozoghatunk jobbra-balra, le-föl, rendezhetjük az adatokat dátum, valamint vonalkód szám szerint a megfelelő nyomógomb megnyomásával. Az alapeset a dátum szerinti rendezettség. A lista nyomtatása gomb lehetőséget ad arra, hogy ne csak képernyőn, hanem nyomtatón is megjelenjen a keresési feltételnek eleget tévő adatok. (2. számú melléklet)

A napi záráslista készítése roppant egyszerű feladat. Nincs más tennivalónk, mint a keresés kezdő és végső időpontjának – a számunkra fontos nap – megadása (pl.: 2008.05.09 – 2008.05.09), de a vonalkód számát, valamint a vevő sorszámát üresen hagyjuk. Így megkapjuk az adott napon eladott összes terméket.

11.3.2. Készletben való keresés

Az előző funkcióval szemben itt a még készleten lévő árucikkek között tudunk keresgetni. Az egész folyamat ugyanúgy működik, mint az eladott áruk keresése eljárás, csak annyi különbséggel, hogy itt nem tudunk megadni vevő sorszámot, mivel ekkor még nem értelmezett ez a paraméter.

Két szempontot tudunk megadni, amelyből a listázási intervallum megadása szintén kötelező, a vonalkód számának a megadása választható. Itt szintén lehetőség van a keresési feltételnek eleget tévő adatok kinyomtatására. (3. számú melléklet)



21. ábra

Ennek a modulnak a részletes bemutatását nem tartom indokoltnak, hisz teljes mértékben ugyanúgy kell használni, mint az eladott áruk keresése funkciót.

11.4. Törzsek

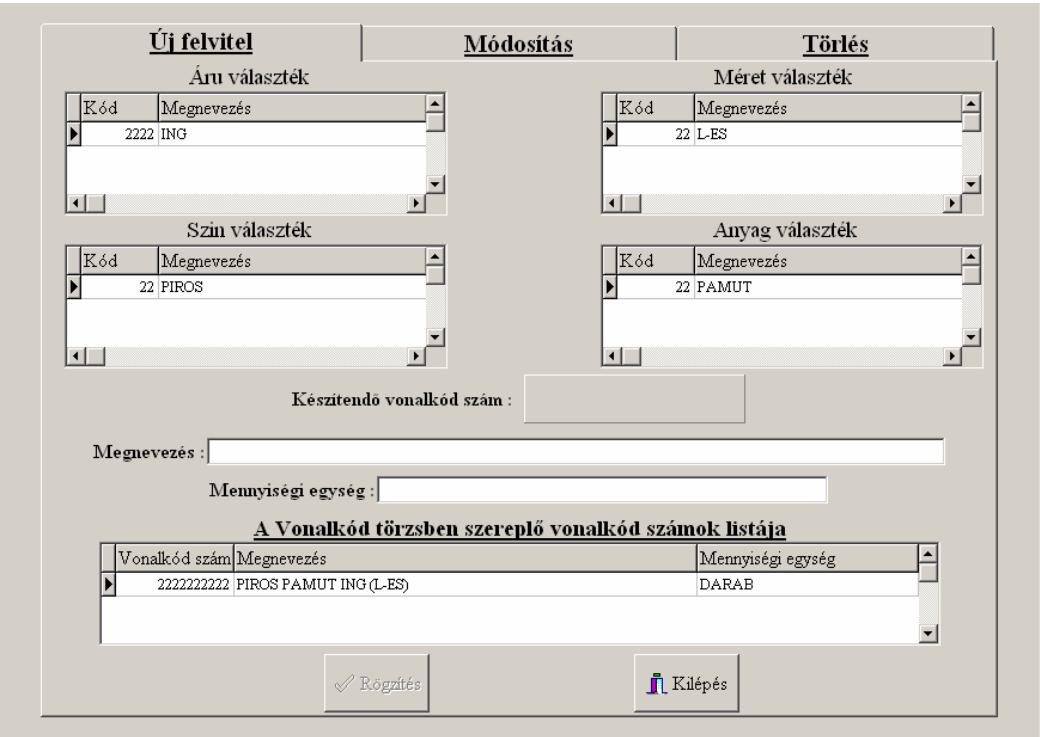
Mivel az adatok tárolásával foglalkozik a szakdolgozati témaként választott programom, ezért mindenképpen nélkülözhetetlen a vonalkód szerkesztéséhez használt egyéb táblák aktualizálása. Vannak olyan adatállományok, melyeket felhasználói oldalról nem kell karbantartani, hisz a szoftver átmeneti adatgyűjtésre, valamint kapcsolótáblaként szolgál.

11.4.1. Vonalkód törzs karbantartása

Mivel a készletbevétel, valamint az eladás vonalkód megadásával történik, ezért fontos, hogy kódot tudjunk készíteni. Ezzel a menüponttal ez lehetséges, sőt a vonalkód törzs karbantartása is megoldott, értem ezalatt a már törzsben lévő azonosítók módosítását, törlését. Az egyes funkciók között mozgás a képernyő tetején lévő lapfülek segítségével történik.

Új felvitel

A képernyőn található négy darab táblázat, melyek segítségével elkészíthetjük a



22. ábra

vonalkódunkat. Az áru, méret, szín és anyag választékból megalkotjuk a terméknek megfelelő azonosítót.

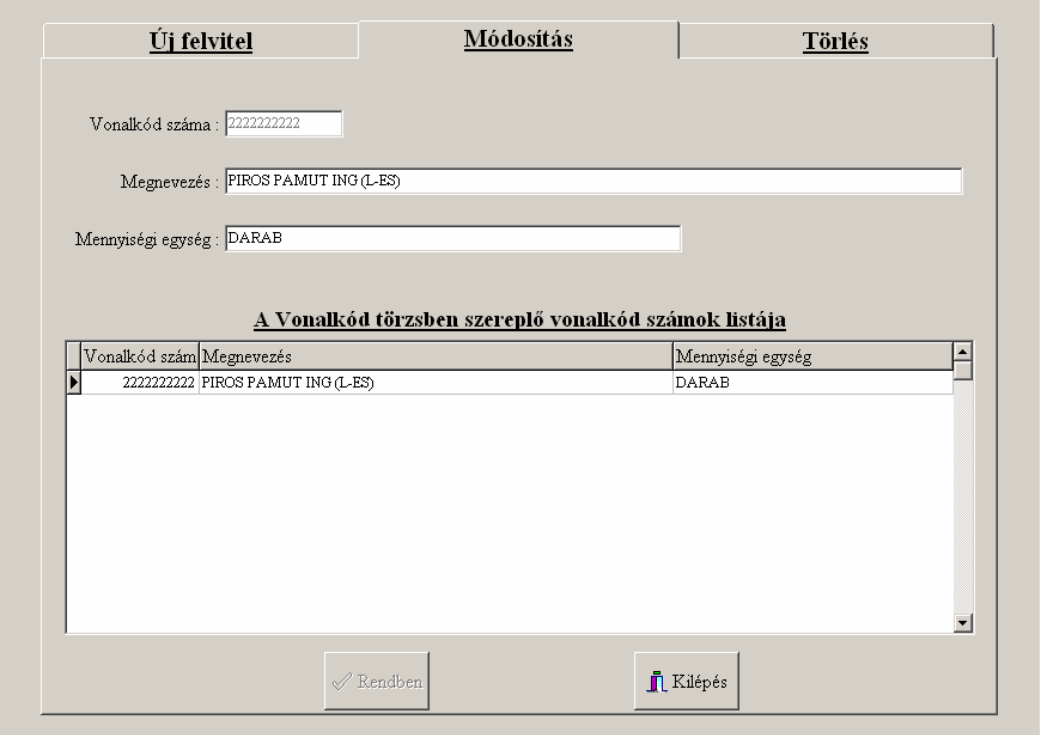
Ez úgy történik, hogy a táblázatokból a megfelelő sorokat kiválasztjuk. A készítendő vonalkód panelon megjelenik az általunk összeszerkesztett szám. A megnevezés mezőben megjelenik a kiválasztott sorok megnevezései összefűzve. Az lesz majd az összeszerkesztett vonalkód megnevezése, melyet természetesen megváltoztathatunk. Meg kell még adnunk egy mennyiségi egységet. A megnevezés és a mennyiségi egység adatait kötelező megadni. Ha mégis elmulasztottuk volna, akkor a rendszer a rögzítés gomb megnyomásakor pirossal beszínezi azokat a mezőket, amelyeket mindenképpen ki kell töltenünk.

A rögzítés gombbal tudjuk felvinni a vonalkód törzsbe az új vonalkód típust. Ha olyan kódot generáltunk, amely már szerepel a törzsben, akkor erről a rendszer üzenet formájában értesíti a felhasználót, és nem rögzíti le a tételt. A képernyő alján található egy táblázat, amelyben láthatóak az eddig már a törzsben szereplő kódok.

23. ábra

Módosítás

Gyakran megtörténik, hogy az új felvitelkor hibát vétünk, és ezt később vesszük észre. Ezzel az opcióval kijavíthatjuk a hibánkat. Nincs más teendő, csak annyi, hogy a listából kiválasztjuk a hibás tételt. A vonalkód számának módosítása nem megengedett, de ezen kívül a megnevezés és a mennyiségi egység megváltoztatható.



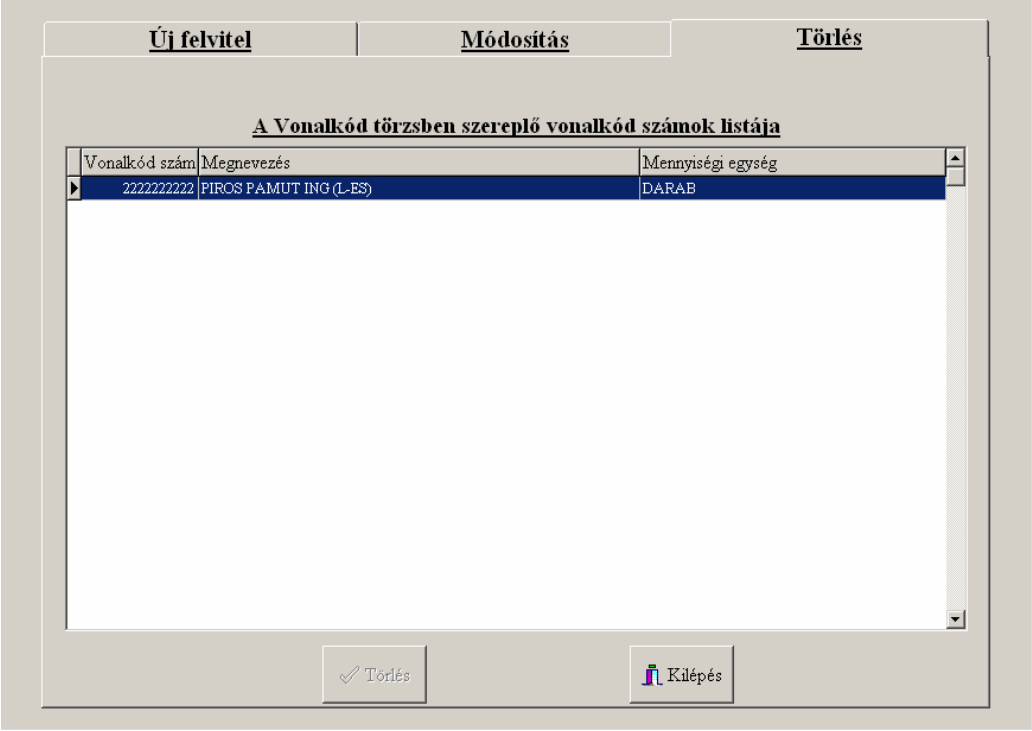
24. ábra

Ha minden hibás adatot kijavítottunk, akkor a rendben gomb megnyomásával ténylegesen is megtörténik a módosítás.

Törlés

Előfordulhat, hogy valamelyik vonalkódot elrontottuk, vagy bármi más okból kifolyólag törölni szeretnénk.

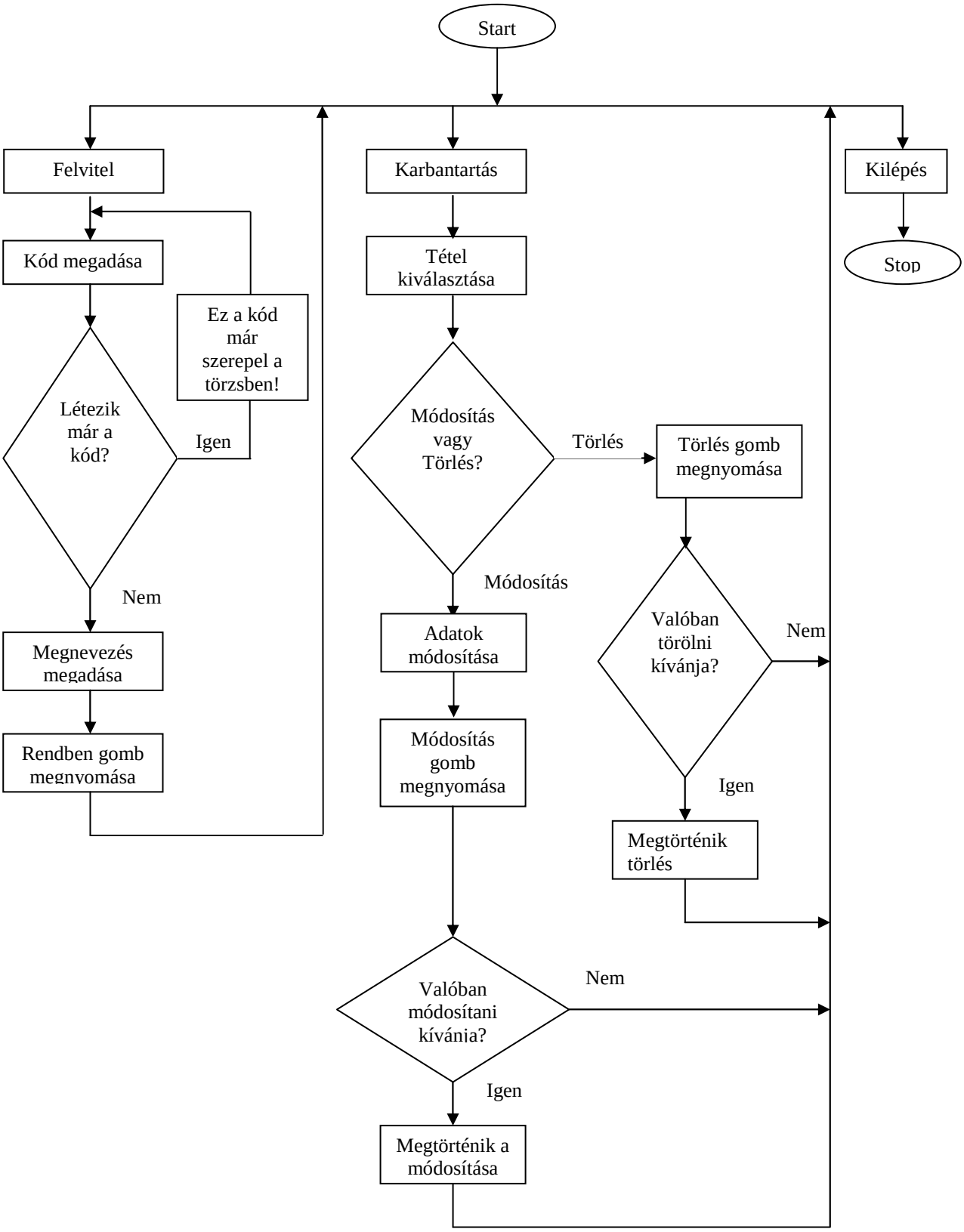
Az előttünk lévő listából kiválasztjuk a törölni kívánt tételt, és megnyomjuk a törlés gombot. Olyan tételt nem lehet törölni, amelyet már egyszer készletbe vettünk, hisz a későbbiek folyamán a rendszer használhatja.



25. ábra

11.4.2. Adatbázisok karbantartása (Áru, Méret, Szín, Anyag)

Az áru, méret, szín, anyag adatbázisok karbantartása teljes mértékben megegyezik, csak más és más táblákon hajtjuk végre a műveleteket, ezért a szakdolgozatomban nem részletezem mindegyik modult, hisz a folyamat egy szisztémára épül. A 26. ábrán láthatjuk a menüpont működési mechanizmusát.


Start
Felvitel
Karbantartás
Kilépés
Stop
Kód megadása
Tétel kiválasztása
Ez a kód már szerepel a törzsben!
Létezik már a kód?
Igen
Módosítás vagy Törlés?
Törlés
Törlés gomb megnyomása
Nem
Módosítás
Adatok módosítása
Valóban törölni kívánja?
Nem
Megnevezés megadása
Módosítás gomb megnyomása
Igen
Rendben gomb megnyomása
Megtörténik törlés
Valóban módosítani kívánja?
Nem
Igen
Megtörténik a módosítása

26. ábra

Első lépésként el kell döntenünk, hogy új adatot szeretnénk felvinni, vagy a már rögzített tételeket akarjuk karbantartani, tehát módosítani, vagy törölni.

Új felvitel

- Az új felvitel nyomógomb megnyomása után lehetőségünk van egy kód beírására, majd a kódhoz tartozó megnevezés megadására. Értelemszerűen, ha a szín kódokat töltjük fel, akkor az azonosítóhoz egy szín megnevezést kell párosítani (pl.: 11-hez piros), ha az anyag kódokat visszük fel, akkor meg az anyagokat kell társítani (pl.: 11-hez farmer) és így tovább a többinél is. Az azonosítók megadásának a szabálya a következő:

- Ahány csillagot látunk a kód mezőben, annyi számjegyet kell megadni. (pl.: ****, akkor 4 számjegyűnek kell lenni a kódnak)

- Nem kezdődhet kód 0-val, tehát nem helyes, ha a következőképpen adjuk meg (pl.: 0111). Értelemszerűen az azonosító első karakterének 1..9 közé eső számnak kell lennie, a kód többi számjegye 0..9 közé eshet.

Minden mező feltöltésének a befejezésekor nyomjunk egy Enter billentyűt. Ha a megnevezést is megadtuk és nyomtunk utána egy Entert, akkor aktívvá válik a Rendben gomb. Ennek a megnyomására rögzítjük az adatainkat.

Karbantartás

Feltétlenül szükséges a tételek módosítása, valamint szükség szerinti törlésük is. Ezeket a feladatokat a következő módon végezhetjük el.

A karbantartás nyomógomb megnyomását követően a táblázatból ki kell választanunk azt a tételt, amelyet módosítani, vagy törölni szeretnénk. A kiválasztás után aktívvá válnak a módosítás és a törlés gombok.

Módosítás

A kiválasztott rekord megnevezését lehet csak módosítani. Szerkesszük át a mező tartalmát, majd ha ez megtörtént, akkor utána nyomjuk meg a módosítás nyomógombot. Ezzel az egyszerű eljárással a hibánkat korrigálhatjuk.

Törlés

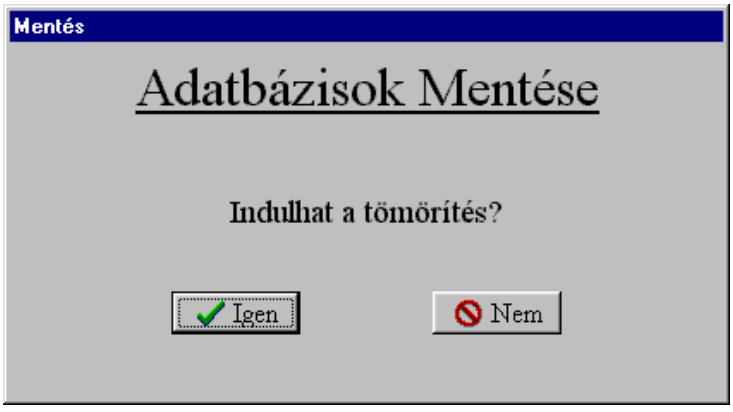
Ez a feladat-végrehajtás még egyszerűbb, mint az előző. Az általunk kiválasztott tételt a törlés gomb megnyomásával lehet eltávolítani az adatbázisból. A program védelmet nyújt azon rekordok számára, amelyek már tartalmazzák ezt a kódot, tehát ha a rendszer valahol használja ezt az azonosítót, akkor nem lehet törölni a kijelölt tételt.

11.5. Szerviz

Mivel a program nagymennyiségű adatok tárolásával foglalkozik, ezért elengedhetetlen feltétel, hogy a rendszer fel legyen készítve a tárolt információk mentésére, valamint az esetleges egyéb meghibásodásból adódóan a mentett adatok visszatöltésére.

11.5.1. Adatbázisok mentése

Az adatbiztonság nélkülözhetetlen, ezért minden rendszer szerves része az adatmentés. A menü meghívását követően rákérdez a program, hogy indulhat-e a tömörítés.



27. ábra

Ha igen, akkor a c:\Bolt\Mentes könyvtárba összetömöríti a c:\Bolt\Data könyvtárban lévő állományainkat. A tömörítést az ARJ nevű programmal végzi a rendszer. A készítendő tömörített állomány neve összetett. A következő példán levezetve megérthető, hogy hogyan is épül fel.

Pl. 08042701.arj

- 08: Évszám (A példában: 2008)
- 04: Hónap (A példában: Április)
- 27: Nap (A példában: 27.)
- 01: Sorszám (A példában: 01)

Értelmezés: Tehát 2008.04.27-én az első mentés!

Ezzel az eljárással egy nap akár 99 db mentést is készíthetünk.

Ha a nem gombot nyomjuk meg, akkor nem indul el a tömörítés, hanem visszajutunk a főmenühöz.

11.5.2. Mentés visszatöltése

A mentett táblák visszatöltésére akkor van szükség, ha az adatbázisaink valamilyen oknál fogva megsérülnek. Ezzel a menüponttal a visszatöltési műveletet lehet elvégezni gyorsan.

A megjelenő listából válasszuk ki azt a mentési dátumot, amelyiket vissza szeretnénk tölteni, és utána nyomjuk meg az igen gombot.

Ezzel elindul a kicsomagolás, és a rendszer visszahelyezi a tömörített állományból a tábláinkat a megfelelő helyre.

Ha a nem gombot nyomjuk meg, akkor nem történik meg a visszatöltés, valamint a visszahelyezés, hanem visszajutunk a főmenühöz.

28. ábra

12.ÖSSZEFOGLALÁS

A dolgozat megírásakor az a cél vezérelt, hogy bemutassam a Borland Delphi-t, és ezen belül a Delphi adatbázis-kezelését. A téma teljes mélységének a feltárására e dolgozat keretei határt szabnak, hiszen olyan nagy anyagról van szó, melyhez könyvek sokasága sem lenne elég.

A szakdolgozat elején pár szót szóltam a programozási nyelvről, majd részletesebben bemutattam az adatbázis kezelést. Egy rövid áttekintő után az adatbázis-kezelési architektúrákról esett szó. Itt felsoroltam a három fő funkcionális egységet, majd egy részletes képet festettem az adatbázis-kezelési architektúrákról.

A következő fejezetben a Borland Database Engine (BDE) és a BDE aliasokról írtam, majd ezek után rátértem a Delphi adatbázis-kezelő komponenseire. Itt sajnos a dolgozat határai arra kényszeríttettek, hogy csak a fontosabb komponenseket említsem meg. Részletesebben az adatelérési és az adatmegjelenítési komponensek egy kis csoportjáról írtam. Mivel az adatokat nem csak a képernyőn szeretjük látni, hanem valamilyen papír formában is, ezért fontosnak tartottam, hogy a jelentés-készítésről se felejtkezzem meg. Egy részletesebb leírás keretein belül vázoltam fel, hogy az adatok nyomtatott formában történő megjelenítésének milyen eszközei vannak. A szakdolgozatomhoz készített szoftver a mellékelt CD-n installációs formában található meg, ezért az installáció lépéseiről is készítettem egy részletesebb beszámolót.

A szakdolgozatom második felében bemutatásra került egy bolti készletnyilvántartó szoftver. Felvázoltam a rendszer által használatos táblákat, valamint a felépítésüket. Tovább haladva bemutatom, hogy a már kész szoftvert miként kell üzembe helyezni, valamint milyen fontos beállításokat kell elvégezni a program jó működése szempontjából.

Az utolsó részben megismerhetjük a menüpontokat részletesen, a megfelelő programképek bemutatásával.

A rendszer Delphi fejlesztői környezetben készült, és törekedtem arra, hogy könnyen kezelhető, érthető legyen egy laikus számára is.

Mivel egy programot soha nem tekinthetünk késznek, ezért ezt a szoftvert is lehet továbbfejleszteni. A későbbiek folyamán több felhasználós rendszer kialakítása lesz szükséges, valamint a felhasználó által kért módosítások, bővítések beépítése.

IRODALOMJEGYZÉK

1. **Bálint Dezső**
   Adatbázis-kezelés
   Talentum Kft. Budapest 1998

2. **Baga Edit**
   Delphi másképp…
   Baga Edit 1998

3. **Marco Cantù**
   Delphi 5 Mesteri szinten I-II. kötet
   Kiskapu Kft. 2000

4. **Kuzmina Jekatyerina**
   Dr. Tamás Péter
   Tóth Bertalan
   Programozzunk Delphi 7 rendszerben!
   ComputerBooks 2007

5. **Szelezsán János**
   Adatbázisok, LSI Oktatóközpont, Budapest

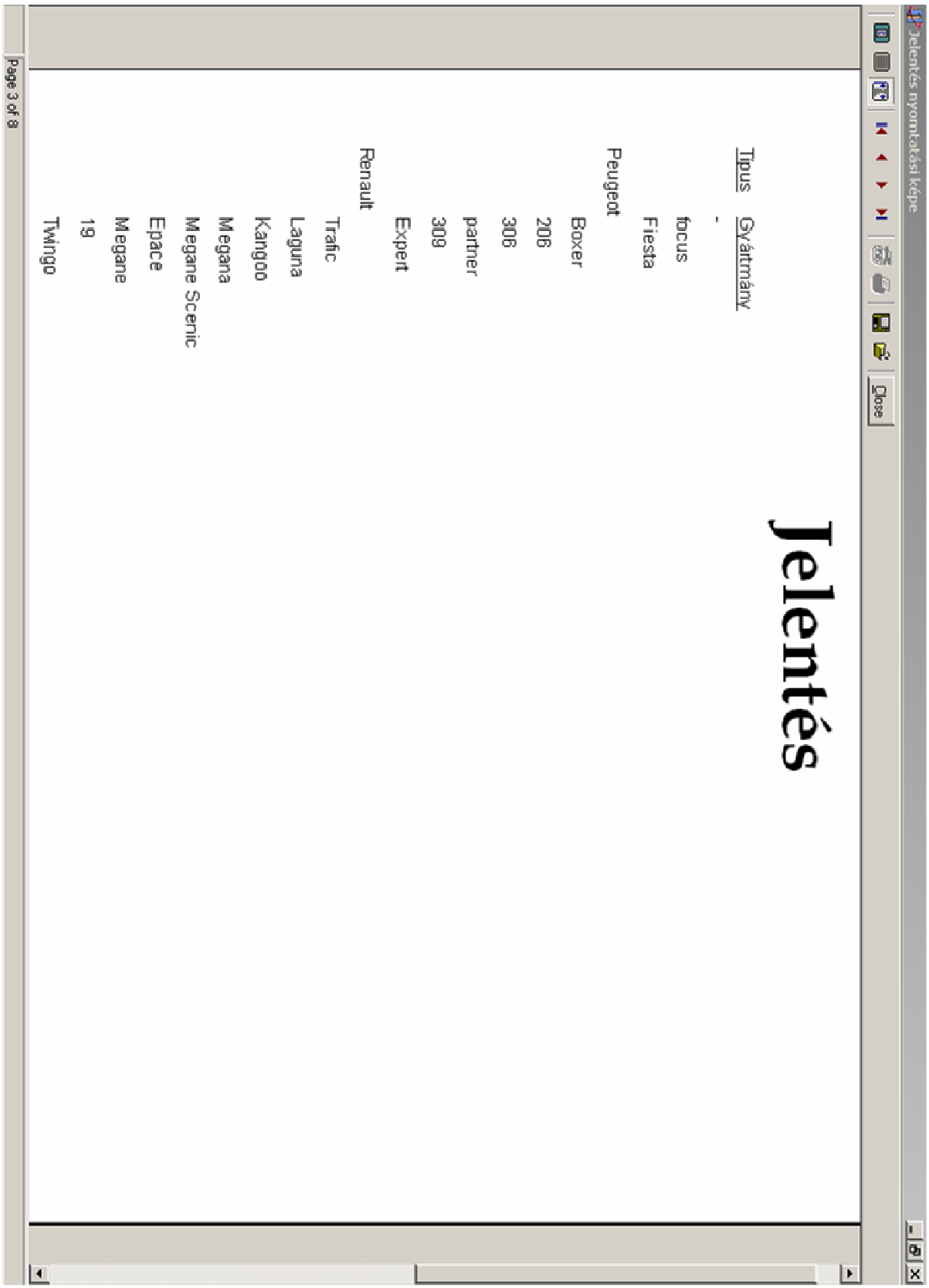
6. **Thomas Binzinger**
   Delphi, Kossuth Kiadó, 1998

7. **Gray Cornell**
   Delphi Tippek és Trükkök, Panem Kft., 1997

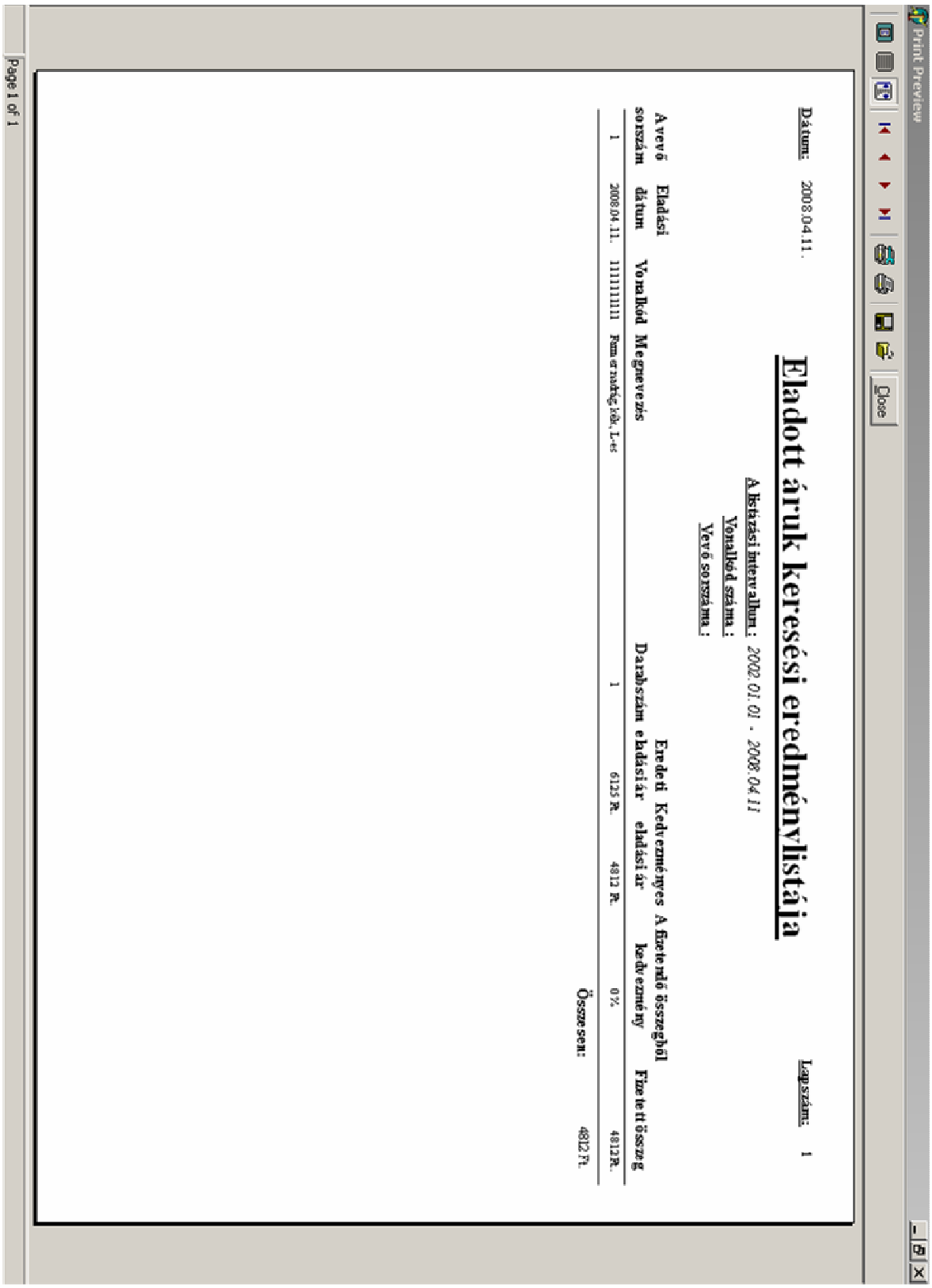
8. **www.borland.hu**

MELLÉKLET

1. SZÁMÚ MELLÉKLET



2. SZÁMÚ MELLÉKLET



3. SZÁMÚ MELLÉKLET